uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería Informática

2019-2020

Trabajo Fin de Grado

"Aplicación de watermarking para la detección de spam"

Sofía Najarro Almaraz

Tutor

Lorena González Manzano

Colmenarejo, Julio de 2020

AGRADECIMIENTOS

Estos cuatro años de carrera han sido una experiencia muy enriquecedora tanto a nivel personal como profesional. Tengo la necesidad de agradecer a todas aquellas personas que han estado a mi lado en este desafiante camino.

A nivel profesional, tengo que agradecer a profesores dedicados como mi tutora Lorena por animarme, tener paciencia y compartir su pasión. Esto tampoco habría sido posible sin mis solidarios y brillantes compañeros.

A nivel personal, quiero agradecer en primer lugar a mi familia por apoyarme. En especial a mi madre, sin ella no hubiera llegado lo lejos que he llegado y no sería quién soy hoy en día. No puedo olvidarme de mi hermana por aguantarme y creer siempre en mí. Estoy eternamente agradecida con estas dos admirables mujeres. También, quiero agradecer a mi novio Álvaro por su apoyo incondicional en todo momento y por hacerme ver que mis defectos se pueden mejorar e incluso transformarse en mis virtudes. En segundo lugar, quiero agradecer a mis amigos por sus risas y alegría en mis peores momentos y por siempre ver todo mi potencial.

Gracias a las quejas de correos de spam de mi madre y la ayuda constante de mi tutora por aplicar mi pasión por la Ciberseguridad a este problema, surgió la idea de este proyecto.

Por último, quiero agradecer a mi abuela Rosa por ser mi principal fuente de inspiración para luchar por mis sueños y nunca rendirme. Siempre estará en mi corazón.

RESUMEN

Actualmente, la comunicación entre los millones de personas de todo el mundo se realiza mediante medios digitales, entre ellos el correo electrónico. Tanto en la sociedad en general como a nivel empresarial, el uso del correo electrónico está muy extendido. En concreto, especialmente en lo relativo a las empresas, es importante que este medio sea seguro, fiable y de confianza, pues mucha de la información intercambiada es sensible. Este hecho unido a los continuos ciberataques, por ejemplo, comenzando por campañas de spam, hace que sea necesario buscar una solución al problema.

En concreto, en este trabajo se desarrolla una aplicación para la detección de spam mediante la verificación de la autenticación del emisor del correo, utilizándose para ello un algoritmo de watermarking. Esto supone una protección frente a un problema muy común, como es la recepción masiva de correos spam en empresas. Para este proyecto, se selecciona un tipo de algoritmo de watermarking específico llamado Zero-Watermarking. Los clientes y las empresas intercambian correos electrónicos, de modo que cada uno de ellos debe añadir una marca de agua para asegurar que los emisores son quien dicen ser y no fuentes fraudulentas. Este proceso de verificación lo realiza una entidad externa.

Palabras claves: correo electrónico, spam, autenticación, Zero-Watermarking, marca de agua.

ABSTRACT

Currently, communication among millions of people around the globe is done via email. Both individuals and companies use it. Especially regarding companies, it is important that the exchange of information stays safe, reliable, and trustworthy. This fact together with the cyber-attacks evolving and upgrading, for example, starting with spam campaigns, makes it necessary to find a solution to this problem.

This project develops an application to detect spam by verifying the authentication of the sender of the email, using a watermarking algorithm for this. This can solve a major problem, the massive reception of malicious spam emails in companies. For this project, a specific type of watermarking algorithm called Zero-Watermarking is selected. The algorithm used will make sure that every time a user sends an email, there will be a watermark added to the message. The authenticity of the email can be verified thanks to the watermark previously generated. This verification is carried out by an external entity.

Key words: email, spam, authentication, Zero-Watermarking, watermark.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE FIGURAS DEL ANEXO

ÍNDICE DE TABLAS

ÍNDICE DE TABLAS DEL ANEXO

1. INTRODUCCIÓN

Para definir este proyecto es importante entender el contexto, las motivaciones que han impulsado esta idea y los objetivos propuestos. También, se detalla la estructura del documento para visualizar mejor globalmente los contenidos a tratar.

1.1. Contexto

El servicio del correo electrónico o e-mail surgió antes incluso de la aparición del Internet. En 1965 se empezó a utilizar y en 1971, la *arroba* de Ray Tomlinson emergió [1]. Este servicio era fundamental para las comunicaciones entre usuarios. A pesar de los avances considerables en la tecnología y en los medios de comunicación, el correo electrónico se sigue utilizando como el canal preferido de comunicación corporativa [2]. Aunque las redes sociales se utilizan a diario por usuarios de todo el mundo, su innovadora funcionalidad de comunicación como mensajería instantánea incluida con muchas más funcionalidades que el que ofrece el e-mail, no son rivales para la mensajería online convencional. Los usuarios prefieren las redes sociales mientras que las empresas prefieren utilizar el e-mail [2].

Al principio, su seguridad estaba garantizada, pero a medida que Internet se ha ido ampliando y la información masificando, los *spammers* han sabido cómo sacarle provecho a la situación. Estos mensajes de publicidad engañosa y fraudes son enviados a diario a los clientes y empleados de empresas del mundo. Según un estudio de duración de un año de Google en 2016 [3], se ha descubierto que el ciberataque más común y peligroso en la red es el ataque *phishing*, ya que, mediante el engaño intentan extraer datos claves del usuario. La técnica preferida es el uso de dominios falsos [4], ya que, es muy difícil de detectar para el usuario. Cada vez hay más estafas en Internet y a su vez, las empresas notifican a sus clientes de la prevención que deben tomar a cabo y toman medidas en sus medios al respecto como son la actualización de los sistemas manteniéndolos libres de virus, concienciar a los clientes, utilizar contraseñas robustas...[5]

Pero al igual que los criminales cibernéticos mejoran sus técnicas de fraude, las empresas no son capaces de controlar todos los mensajes engañosos. Solo con introducir tus datos en una página web se está poniendo en peligro tu seguridad frente a los *spammers*. Además, los servicios de mensajería instantánea actuales no son capaces de saber si el remitente del correo es de verdad quien dice ser. Una solución propuesta por las empresas es la firma electrónica para demostrar su veracidad.

Este proyecto se centra en la creación de un prototipo de aplicación para detectar correos spams. No solo detectarlos sino, además, ofrecer un canal seguro donde las empresas puedan garantizar que el correo no ha sido manipulado ni se ha suplantado una identidad. Este prototipo utiliza un algoritmo de *watermark* o marca de agua como firma. Esta aplicación mejora la seguridad de los correos electrónicos en la identificación de mensajes de spam.

1.2. Motivación

Esta propuesta se origina en mi interés por la rama de ciberseguridad y en la curiosidad de explorar este campo antes de realizar el Máster de Ciberseguridad.

La creación de una aplicación de *watermarking* para la detección de spam surgió de las recientes quejas de envíos excesivos de mensajes spam en los correos electrónicos en mi círculo social. Los gestores de correos electrónicos no son capaces de detectar todos los correos spam, ya que, estos cada vez pasan más desapercibidos. Además, el correo electrónico es usado ampliamente en todo el mundo, sobre todo lo utilizan las empresas y sus clientes. Normalmente, el intercambio de información es confidencial. Si se manipulara o se hiciera un mal uso de ese mensaje, se podría poner en peligro la seguridad de ambos. Por ello, las motivaciones principales de esta propuesta son velar por la seguridad y la información de los correos enviados entre empresas y clientes y detectar aquellos que son correo spam, ya que, es un problema bastante común actualmente donde las consecuencias pueden ser muy graves.

1.3. Objetivos

El objetivo principal es ayudar a los usuarios en la identificación de emails fraudulentos y aumentar su confianza en los sistemas de mensajería online.

Para conseguir el objetivo anterior, se plantea la búsqueda de un algoritmo criptográfico para que los correos estén firmados y se pueda saber quién los ha escrito. Además, se quiere implementar una aplicación prototipo para la detección de spam funcional.

1.4. Estructura del documento

Este documento se divide en 9 capítulos y el anexo

En el Capítulo 1. Introducción, se explica el origen de los correos electrónicos y la aparición de las técnicas fraudulentas, en especial, de los correos spam. Además, se define en qué consiste este proyecto complementándolo con las motivaciones y objetivos propuestos.

En el Capítulo 2. Conocimientos previos, se explican las competencias necesarias anticipadas para que el lector entienda este documento.

En el Capítulo 3. Estado del arte, se hace una investigación exhaustiva de los trabajos relacionados que hagan algo similar al proyecto. Se demuestra la novedad de la propuesta.

En el Capítulo 4. Análisis, se expone el estudio tecnológico llevado a cabo para desarrollar la aplicación más adelante, se hace un análisis de requisitos y de sus casos de uso.

En el Capítulo 5. Diseño, se describen las decisiones tomadas sobre el planteamiento de la aplicación limitadas por el marco regulador. El *marco regulador* es

fundamental, ya que, consiste en el análisis de la legislación aplicable. Además, se detalla el esquema del sistema con los componentes software, sus modelos de clase y sus diagramas de secuencia.

En el Capítulo 6. Implementación, se especifican los conceptos más relevantes del código desarrollado y se muestran los esquemas resumidos de las pantallas de la aplicación y de las funcionalidades principales para una mejor comprensión del código.

En el Capítulo 7. Evaluación, se define cómo se van a realizar las pruebas mediante un plan de pruebas y su verificación de que el sistema funciona correctamente.

En el Capítulo 8. Gestión de proyecto, se muestra un programa de la planificación de las diferentes tareas para una mayor dirección del proyecto. También, se estiman los costes para extraer un presupuesto final para el cliente. Por último, se destaca el *impacto socioeconómico* que aportará más valor al trabajo final de carrera.

En el Capítulo 9. Conclusiones y mejoras futuras, se explican los resultados obtenidos al realizar este proyecto, las dificultades encontradas, lo aprendido a nivel personal y profesional y se sugieren posibles modificaciones en la posteridad para un mejor funcionamiento.

En el Anexo: Manual de Uso, se guía al usuario a través de la aplicación implementada. Se muestran los requisitos previos necesarios, las instrucciones de instalación y las pautas de ejecución.

En el Anexo: Plantillas, se exponen las tablas de las plantillas utilizadas para los requisitos, los casos de uso y las pruebas de aceptación.

2. CONOCIMIENTOS PREVIOS

Es necesario explicar previamente algunos conceptos relevantes que van a aparecer con frecuencia a lo largo del documento.

La esteganografía, del griego **steganos**(oculto) y **graphos**(escritura), es la ocultación de información en un canal encubierto evitando el descubrimiento del mensaje escondido [6]. No nació con la informática. Es un arte que lleva usándose desde que el hombre tuvo la necesidad de ocultar mensajes.

La esteganografía clásica se basa en el desconocimiento del canal encubierto específico que se está usando. Se encuentran diversos ejemplos a lo largo de la historia. Por ejemplo, se cortaba el pelo del animal o el del esclavo, se grababa debajo un mensaje y se esperaba a que el pelo creciera para enviar la información sensible [7]. Para descifrar el mensaje se cortaba el pelo y se veía el mensaje oculto. También, se usaba zumo de limón en un papel como tinta invisible para esconder lo escrito [7]. Para descifrarlo se aplicaba calor, por ejemplo, usando una llama generando combustión para que apareciese el mensaje. Incluso durante la Segunda Guerra Mundial, en los periódicos se perforaban letras que al exponerlas a la luz formaban un mensaje [6].

Para entender mejor como funciona, se explica un pequeño ejemplo. Los elementos clave son :

- Mensaje secreto: información a ocultar
- Objeto contenedor o encubridor: elemento de tapadera o camuflaje.
  Ejemplo: Esclavo
- Estego-objeto: objeto contenedor más el mensaje oculto.
  Ejemplo: Esclavo con mensaje oculto debajo de su pelo.
- Guardián: quién monitoriza la comunicación. Hay tres tipos: activos, pasivos o malicioso. Uno pasivo trata de descubrir el algoritmo sin modificar ningún objeto(sólo lectura). Uno activo además modifica el estego-objeto. Uno malicioso puede hacer cualquier de las dos cosas anteriormente descritas (no es realista).
- Estego-función: función estego o técnica para que la tapadera pase desapercibida.
- Estego-clave: opcional para descifrar la información.

En resumen, el mensaje viaja por un canal inseguro y puede ser interceptado por otra persona (guardián). El objetivo es que el guardián desconozca la tapadera o al menos, no pueda descifrarlo. Para descubrirlo se aplica la estego-función a la inversa y la estego-clave si es que tiene [7]. Todos los elementos de la esteganografía se relacionan en el esquema de la figura 2.1.

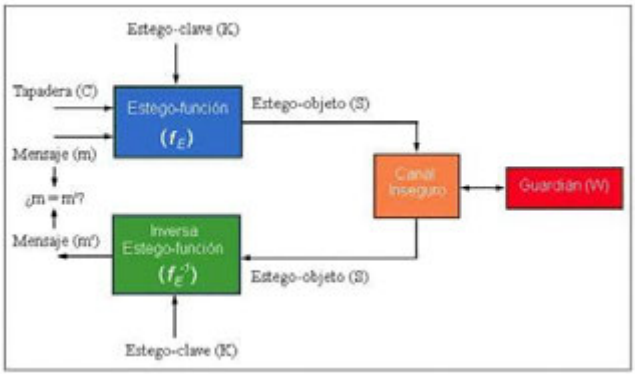


Fig. 2.1. Esquema de la esteganografía [6]

La "*Estego-terna*" está formada por tres elementos: la capacidad, cantidad de información secreta; la seguridad, dificultad de detección del mensaje oculto; la robustez, conjunto de modificaciones posibles antes de perder toda la información encubierta [7]. En la figura 2.2. , hay una representación gráfica de la "*Estego-terna*". Cuanta más información (cantidad), menor robustez y menos desapercibido (más visible). Para una buena ocultación del mensaje es necesario el **equilibro** en la terna.

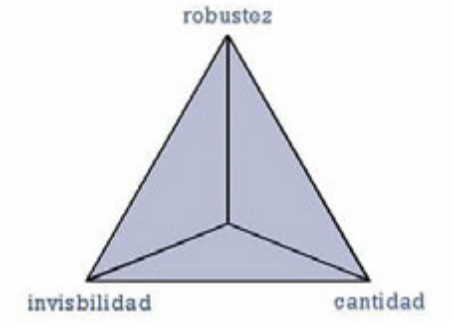


Fig. 2.2. Esquema de la estego-terna[6]

La esteganografía actual funciona parecida a la clásica, pero se aplica a medios digitales. Existen diferentes tipos de esteganografía según el tipo de tapadera: de contenido multimedia, texto, sistemas operativos y ficheros… Además, existen diferentes técnicas de esteganografía, este documento se centrará en el *watermarking* en textos.

El *watermarking* o marca de agua digital consiste en esconder un mensaje en un objeto digital, en este caso en los textos de los correos electrónicos. Se suele utilizar para vincular el autor al objeto, es decir, como técnica para sellar la autenticación.

La autenticación es el servicio que verifica la identidad de un usuario. El mecanismo de la firma digital se utiliza para validar la autenticidad e integridad del mensaje [8]. Está compuesta por dos claves, una pública y otra privada. Con la firma digital, se demuestra que el mensaje no ha sido manipulado y que el emisor del mensaje ha sido verificado [9]. En el proceso de firma digital, se realiza una función hash sobre el documento a firmar. Una función hash es un algoritmo matemático que transforma una cadena de datos en otra con una longitud fija. Por último, se cifra la cadena devuelta por la función hash mediante la clave privada [10]. Para descifrar la firma digital es

necesario utilizar la clave pública. Al descifrar se puede verificar la firma digital. Los procesos de firma digital y verificación explicados se muestran en la figura 2.3.

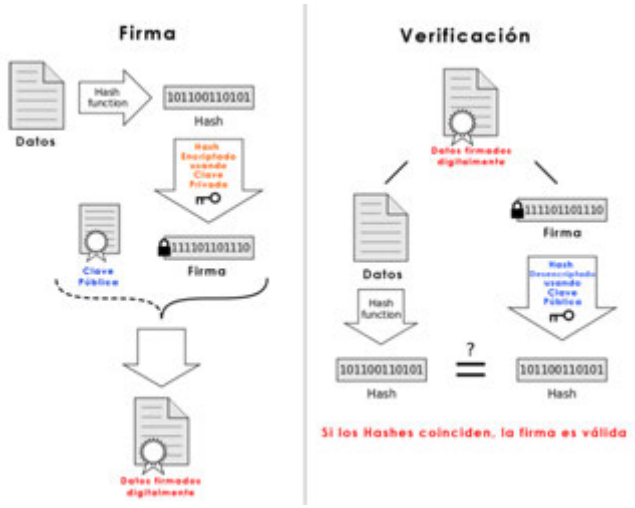


Fig. 2.3. Esquema firma digital [10]

Para reforzar la seguridad de esta técnica, se puede utilizar un elemento más. De las diversas medidas existentes para mejorar la seguridad, existe una entidad llamada *certificate authority* (CA) [13]. Un *certificate authority* (CA) es una empresa externa que valida la autenticación de individuos o empresas. Se generan dos claves criptográficas: una pública y otra privada. Además, se genera un *certificate signing request* (CSR) con la clave pública e información relevante de la empresa como el dominio o correo electrónico [11]. Se envía el CSR al CA. El CA verifica que la información es correcta y firma digitalmente el certificado con la clave privada [11]. Todas estas funcionalidades de CA se muestran en el esquema de la figura 2.4.

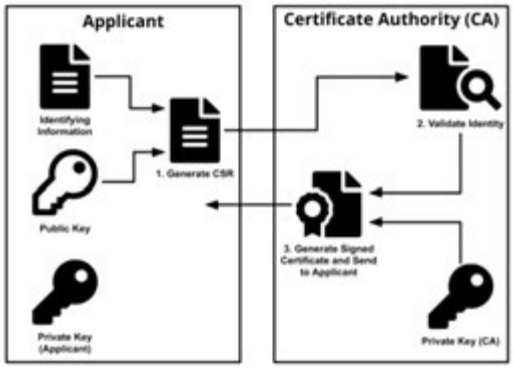


Fig. 2.4. Esquema CA [11]

3. ESTADO DEL ARTE

Para explicar cómo se ha llegado a la solución propuesta es necesario entender su contexto, es decir, todos los trabajos previos que se han realizado y las aplicaciones similares que existen actualmente. Tras ese estudio, se podrá formalizar la propuesta destacando su novedad.

3.1 Trabajos previos

Hay varios trabajos de fin de grado que aplican esteganografía. Los más comunes ocultan un mensaje secreto en una imagen [12]. Por otro lado, la esteganografía en textos se ha tratado más de forma teórica que práctica. Se han seguido desarrollando algoritmos en documentos para la autenticación utilizando la esteganografía [13]. Pero trabajos previos sobre la detección de spam no hay nada en absoluto.

Por otro lado, existen multitud de artículos científicos sobre detección de spam. En un artículo, se detecta el spam mediante las relaciones entre emisor y receptor en Twitter [14]. En otro, se detectan los mensajes fraudulentos mediante correlación, principios de simetría de paquetes y un método estadístico llamado "*Sequential Probability Ratio Test*" [15]. Además, se pueden identificar correos spam mediante el agrupamiento o *clustering* de texto basado en el modelo de espacio vectorial [16].

También existen varios artículos científicos sobre algoritmos de watermarking en textos. En un artículo, se expone un algoritmo de watermarking basado en la clasificación de las palabras y estadísticas de los espacios entre palabras [17]. En otro, el algoritmo de watermaking de texto utiliza un watermarking de la imagen del logotipo del copyright [18]. Por último, un algoritmo de watermarking que utiliza los bits menos significativos (LSB) junto a los bits inversos [19].

Finalmente, para detectar spam se va a utilizar un algoritmo *zero* de marca de agua [13]. A pesar de la existencia de artículos científicos de detección de spam y otros de watermarking en textos, no hay artículos científicos sobre la detección de spam mediante watermarking.

3.2 Aplicaciones similares

En esta sección, se muestran las aplicaciones más parecidas a la solución propuesta para la detección de spam.

DocuSign

Aplicación de firma digital que se enfoca en la autenticación y así evitar fuentes fraudulentas. Existe una extensión de Chrome para poder firmar correos electrónicos directamente desde Gmail [20]. En las figuras 3.1. y 3.2., se puede visualizar su logotipo y su interfaz gráfica.

Fig. 3.1. Logo de DocuSign [20]

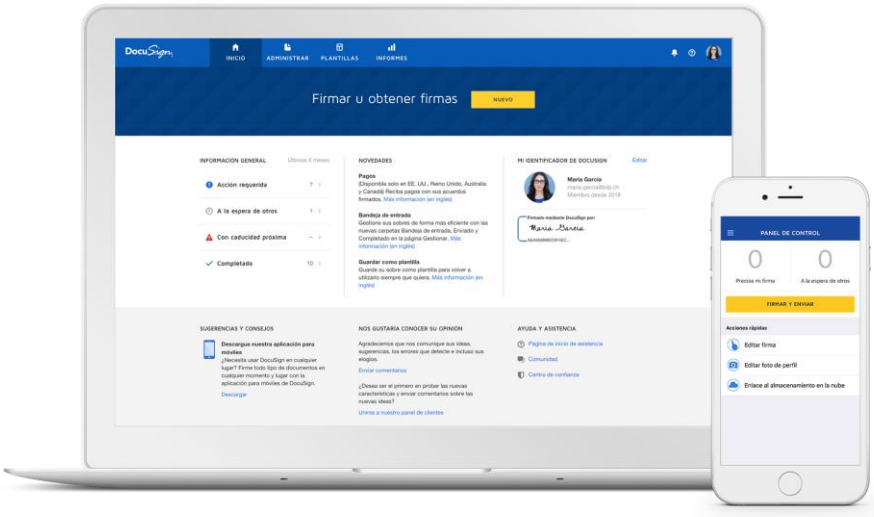


Fig. 3.2. Interfaz de DocuSign [20]

Mailinblack Protect

Mailinblack Protect es un producto dirigido para empresas y clientes que bloquea y filtra los correos spam automáticamente gracias a su base de datos y servidores que funcionan con modelos de Inteligencia Artificial. Además, tiene otras funcionalidades como detectar intentos de phising o spear-phising y anuncios no deseados. Según Mailinblack, una personar tardaría una media de 5 horas en revisar su correo spam, pero con su herramienta software tardarían 40 minutos al día [21]. En las figuras 3.3. y 3.4., se puede visualizar su logotipo y su interfaz gráfica.

Fig. 3.3. Logo de Mailinblack [21]

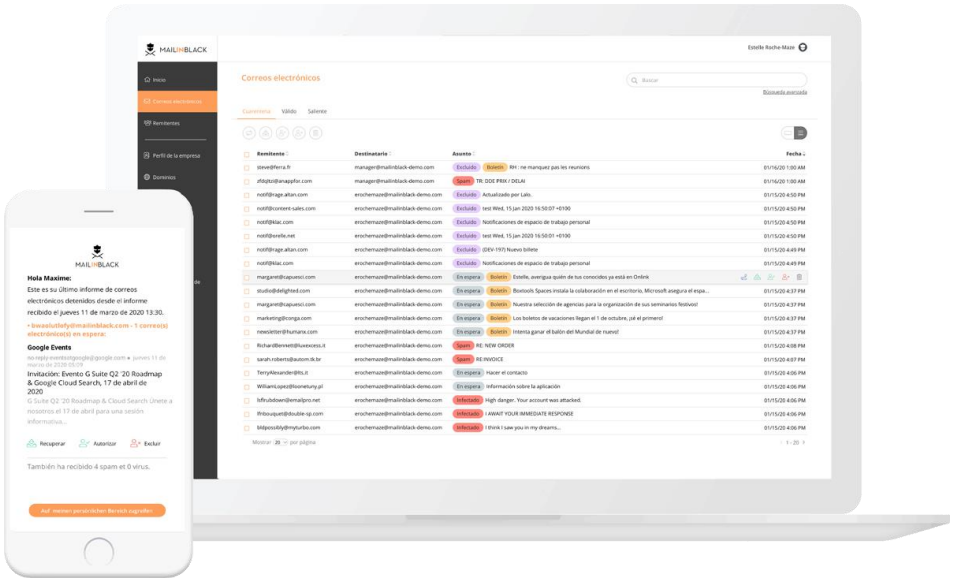


Fig. 3.4. Interfaz de Mailinblack Protect [21]

Claranet

Servicio de antispam y antivirus para correos electrónicos. Ofrece un filtrado automático y garantizado. Usa tecnologías de detección de patrones de spam a tiempo real [22]. En las figuras 3.5. y 3.6., se puede visualizar su logotipo y su esquema de servicios que ofrece.

Fig. 3.5. Logo de Claranet [22]

Fig. 3.6. Esquema de funcionalidades Claranet [22]

3.3 Novedad de la propuesta

En primer lugar, como la mayoría de los trabajos previos tratan sobre el caso práctico de ocultación de datos en textos, en este trabajo se plantea su aplicación para la detección de correos spam y se ha decidido utilizar el algoritmo *zero* [13], generando una marca de agua zero en los textos de los correos electrónicos para verificar la autenticación de forma práctica en una aplicación software. La idea en sí no se ha desarrollado previamente, lo más similar de forma teórica.

En segundo lugar, la novedad de la propuesta también reside en el conjunto de ideas mejoradas de las aplicaciones ya existentes. En la tabla 3.1, se esquematizan para una mejor compresión.

TABLA 3.1. MEJORAS DE LAS APLICACIONES SIMILARES

Aplicaciones similares	Características para la solución propuesta
DocuSign	+ Plugin para Gmail (Mejor accesibilidad) + Firma digital pero mejorada con esteganografía derivando en una marca de agua. (Protección oculta)
Mailinblack Protect	+ App que detecta spam de forma automática para empresas y sus clientes. + Sustitución del uso de filtros automáticos para la detección de spam por el uso de un algoritmo llamado watermark o marca de agua para verificar la autenticación. (Más preciso y seguro)
Claranet	+ Los patrones a tiempo real pueden fallar, pero si se comprueba cada correo individualmente se reducen errores.

Al utilizar las aplicaciones explicadas en el apartado anterior, los usuarios se ahorrarían horas identificando los correos fraudulentos, ya que, sería un proceso automático. Incluso si únicamente se optimizara la bandeja de entrada del correo electrónico mediante filtros antispam y otras medidas manuales disponibles en la configuración [23].

La solución propuesta sugerida economizará las horas empleadas en detectar correos spam por los usuarios, no es necesario configurar manualmente los filtros como ocurre en Gmail. La aplicación más parecida a la solución propuesta es la de *Mailinblack Protect*. Aunque tiene más funcionalidades y usa tecnología más avanzada, la idea de la solución propuesta es más sencilla, es decir, se centra únicamente en detectar correos spam. Además, al utilizar el *watermark* o marca de agua se mejora la precisión de identificación de correos spam. Los bloqueos y filtros automáticos de *Mailinblack* pueden fallar [24], pero la firma con el *watermark* o marca de agua es menos propensa a errores debido a que se está comprobando individualmente cada correo electrónico. Como la marca de agua es una técnica de la esteganografía, es menos detectable que la firma digital de la aplicación DocuSign.

En resumen, el ahorro de horas para el usuario debido a su proceso automático, su sencillez y su enfoque a la seguridad, hacen que esta idea sea una propuesta con un potencial significativo. Además, no se ha realizado antes ninguna aplicación usando esteganografía para identificar correos spam.

4. ANÁLISIS

Antes de abordar el problema hay que realizar algunas consideraciones. Para crear una aplicación de *watermarking* para la detección de spam es necesario seleccionar un lenguaje y un entorno de desarrollo. Como se van a gestionar muchos correos y empresas, es necesario un uso adecuado del almacenamiento y un acceso rápido a las bases de datos. Aunque se trate de un prototipo la interfaz gráfica también es importante. Por ello, será necesario escoger una herramienta gráfica sencilla, cómoda y con diversos recursos. Para tener un modelo más concreto, se realiza un estudio tecnológico, se extraen los requisitos y se definen los casos de uso.

4.1. Solución propuesta

Antes de buscar qué tecnologías se van a emplear, hay que definir la idea propuesta en términos generales.

Para poder enviar un correo electrónico es necesario un emisor y un receptor. En este caso, el emisor sería la empresa y el receptor su cliente. Además, es necesario una entidad intermediaria para verificar que el correo enviado pertenece al emisor original y no es un correo spam.

Más concretamente, es necesario crear tres aplicaciones. La aplicación para las empresas se llamará "*AquaSpam*". El servicio principal de AquaSpam será enviar un correo electrónico a su cliente con la firma de *watermarking*. La aplicación para clientes se llamará "*Gmail Add-on AquaSpam*" y estará de alguna manera disponible dentro de Gmail para poder extraer la información y enviar una petición de verificación a la aplicación intermediaria. La aplicación intermediaria llamada "*CA*", tendrá dos bases de datos. Una base de datos se encargará de almacenar las marcas de agua creadas cada vez que una empresa envía un email a su cliente. La otra base de datos se encargará de almacenar las peticiones de verificación solicitadas por el cliente. Además, en la aplicación de CA se podrá verificar cada petición recalculando la marca de agua y comparándola con las almacenadas en la base de datos de las marcas de agua. Por último, se enviará los resultados por correo y se eliminarán de las bases de datos correspondientes para ahorrar espacio de memoria. Todo ello se esquematiza detalladamente en la figura 4.1.

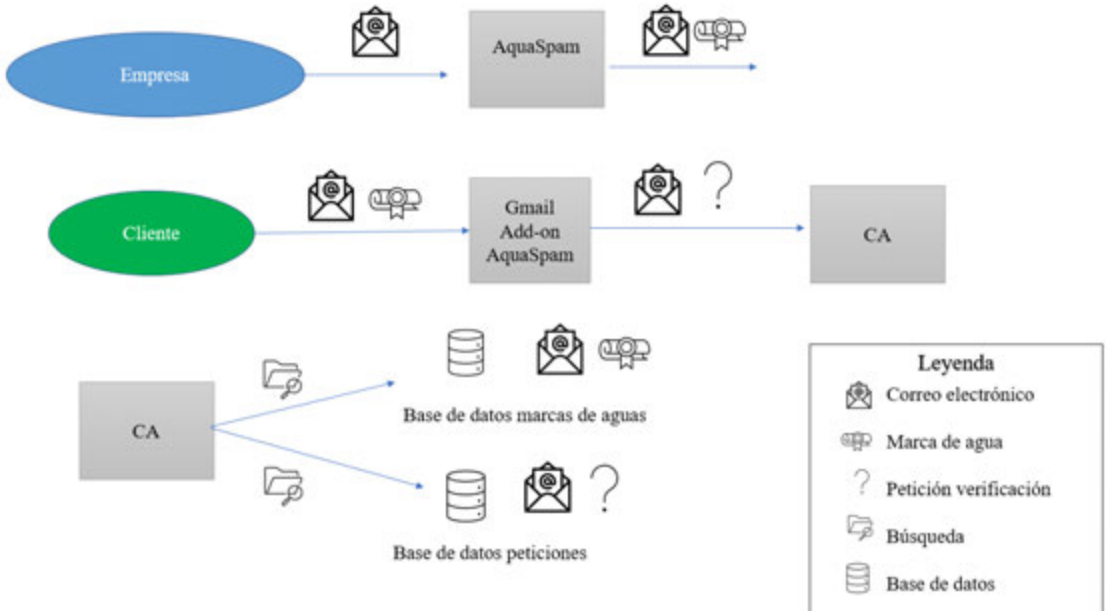


Fig. 4.1. Esquema solución propuesta

4.2. Estudio tecnológico

Este problema se podía abordar de diversas formas. Se podía utilizar cualquier lenguaje, ya que, a lo largo de la carrera se aprenden diversos lenguajes de programación: *Java, C, C++, Python*… Para mantener la innovación de la solución propuesta se decide hacer un estudio de los dos lenguajes de programación más usados de 2019 [25].

Lenguaje	Particularidad	Aplicaciones
1. Phyton	Sintaxis sencilla y fácil de entender Ahorra tiempo y recursos	Inteligencia artificial Aplicaciones Big Data Desarrollo web
2. Java	Orientado a objetos Diseñado para tener dependencias mínimas	Software dispositivos móviles App Android Terminales de venta Cajeros automáticos Internet de las cosas (IoT) Videojuegos Páginas web

Fig. 4.2. Ranking Top 2 aplicaciones más usadas [25]

Según las figuras 4.2. y 4.3., Python parece el más indicado con lo que respecta a la gestión de las bases de datos. En la mayoría de los casos, Java es más rápido, pero menos flexible que Python [26]. Java puede utilizarse en cualquier plataforma con una máquina virtual de Java. La mayoría de los dispositivos la tienen. Java tiene un mayor potencial para aplicaciones Android. Por tanto, ambos son lenguajes muy adecuados para la solución propuesta pero *Java* se adapta más por su rendimiento, rapidez y portabilidad.

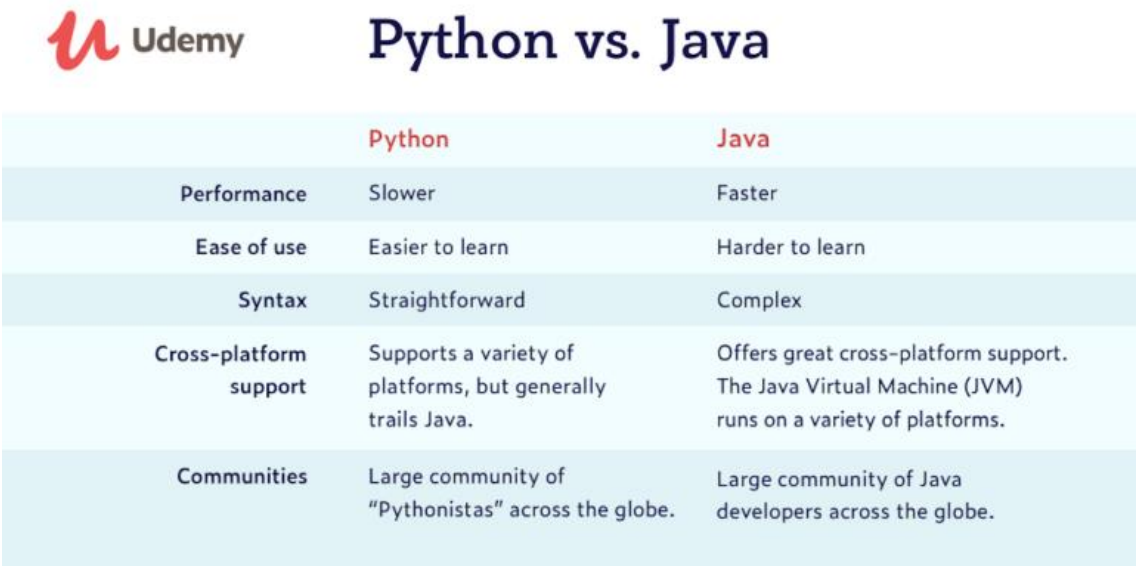


Fig. 4.3. Tabla comparativa Pyhton vs. Java [26]

Una vez seleccionado el lenguaje, hay que buscar un entorno de desarrollo integrado compatible con el lenguaje seleccionado. Para Java, los entornos más conocidos son NetBeans y Eclipse [27].

A pesar de tener numerosas ventajas como el desarrollo de aplicaciones web dinámicas y de servidor, *NetBeans* era muy lento e incompatible con mis equipos informáticos. Pero *Eclipse Photon* combinada con la librería gráfica *Swing* para la interfaz gráfica funcionaba perfectamente en mis ordenadores. La sencillez del *framework Swing* de añadir *widgets* como son los elementos *JFrame* y *JButtons* con código e incluso arrastrar los componentes (arquitectura modelo-vista-controlador) fue decisivo. *Eclipse* puede ser un poco lento con programas grandes debido al consumo excesivo de recursos, pero los programas de las tres aplicaciones van a tener un tamaño no muy extenso y, además, sus códigos serán los más eficientes y eficaces posible.

Para almacenar los correos es necesario una opción sencilla, rápida y eficiente. Tras investigar los diferentes tipos de estructuras, la estructura dinámica eficiente ***HashMap*** encaja con el problema a la perfección. Destaca por la rapidez de inserción de elementos. Cada entrada es una combinación de una clave y un valor. Esta decisión va a ahorrar mucho tiempo de compilación en el código.

Además, va a ser necesario acceder a los correos electrónicos desde *Eclipse*. Tras indagar en las fuentes actuales, cabe destacar una librería externa que permite entrar en un correo de *Gmail* y realizar diversas funcionalidades como es la de enviar un email. Se llama "*javamail-1.4.7*".

Por último, hay que encontrar una solución para el cliente a la hora de enviar una solicitud de verificación desde el correo de Gmail. Tras buscar información, la opción utilizada fue crear un *add-on* de *Gmail* mediante la plataforma de desarrollo de aplicaciones *Google Apps Script* de *G Suite* en lenguaje *JavaScript*. La principal ventaja

es la comodidad y su accesibilidad, ya que, se añadiría al correo de Gmail como un plugin y al alcance en todo momento.

En la tabla 4.1, se expone la solución propuesta resumida de lo explicado anteriormente.

TABLA 4.1. SOLUCIÓN PROPUESTA DEL ESTUDIO TECNOLÓGICO

Solución propuesta				
Aplicación	**Lenguaje programación**	**Herramienta software**	**Biblioteca gráfica**	**Librerías externas**
AquaSpam	Java	Eclipse Photon	Swing	JavaMail v. 1.4.7.
CA	Java	Eclipse Photon	Swing	JavaMail v. 1.4.7.
Gmail Add-on AquaSpam	JavaScript	Google Apps Script de G Suite	(Ya incluida)	----------

4.3. Análisis de requisitos

Para poder implementar cualquier componente software, es fundamental especificar las funcionalidades del sistema junto a sus restricciones. Todo ello se recoge en la Especificación de Requisitos del Software (SRS). Existen dos tipos de requisitos fundamentales: funcionales y no funcionales.

Los requisitos funcionales describen el funcionamiento y los servicios del sistema. Deben ser completos, es decir, deben cumplir todas las expectativas del cliente y ser consistentes.

Los requisitos no funcionales definen las restricciones del sistema, funciones o sus propiedades como el rendimiento, seguridad y disponibilidad.

Al tratarse de tres aplicaciones hay que organizar los requisitos en tres categorías: AquaSpam (App empresa), Gmail Add-on AquaSpam (App cliente) y CA (App intermediario).

Se va a utilizar la plantilla de la tabla A.1 para definir los requisitos. Los requisitos de software se muestran en las tablas 4.2, 4.3 y 4.4, una por cada categoría anteriormente descrita.

TABLA 4.2. REQUISITOS SOFTWARE DE AQUASPAM

Requisitos de Software de AquaSpam				
Tipo: Funcional				
Categoría: Inicio de Sesión				
Id	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RFA-01	Iniciar sesión	Los usuarios podrán iniciar sesión introduciendo: un correo electrónico de Gmail y una contraseña.	Alta	Alta
RFA-02	Cancelar inicio de sesión	El sistema permitirá cancelar el inicio de sesión.	Alta	Media
Categoría: Envío de correo				
RFA-03	Enviar mensaje con algoritmo	El sistema será capaz de enviar correos electrónicos aplicando un algoritmo al rellanar unos campos: la dirección de correo destinataria, el asunto y el mensaje de texto.	Alta	Alta
RFA-04	Borrar campos del mensaje	El sistema permitirá borrar todos los campos del mensaje preparado para enviar.	Alta	Baja
RFA-05	Cancelar mensaje	El sistema permitirá cancelar la creación del mensaje a enviar.	Alta	Media
Categoría: Notificación				
RFA-06	Notificar envío del mensaje	El sistema debe notificar al usuario cada vez que se envía un mensaje.	Alta	Alta
Categoría: Algoritmo				
RFA-07	Buscar palabra más repetida	El sistema internamente permitirá buscar la palabra más repetida en el texto.	Alta	Alta
RFA-08	Definir *kw* o clave del algoritmo	El sistema internamente podrá usar la palabra más repetida encontrada como *kw* o clave para el algoritmo.	Alta	Alta
RFA-09	Definir *watermark* del	El sistema internamente permitirá almacenar la frecuencia de las palabras anteriores y	Alta	Alta

	mensaje	posteriores al *kw* en un watermark integrado en el mensaje junto al *kw*.		
Tipo: No funcional				
RNFA-01	Idioma	La aplicación únicamente mostrará la información en inglés.	Media	Alta
RNFA-02	Mensajes de error	El sistema debe proporcionar mensajes de error para informar al usuario.	Alta	Alta
RNFA-03	Validar datos introducidos	El sistema debe validar los datos introducidos por el usuario.	Alta	Alta
RNFA-04	Seguridad	Los usuarios deberán iniciar sesión para poder usar el sistema.	Alta	Alta
RNFA-05	Seguridad	La contraseña debe almacenarse en hash.	Alta	Alta

TABLA 4.3. REQUISITOS SOFTWARE DE GMAIL ADD-ON AQUASPAM

Requisitos de Software de Gmail Add-on AquaSpam				
Tipo: Funcional				
Categoría: Funciones básicas				
Id	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RFB-01	Seleccionar correo electrónico	El sistema permitirá seleccionar un correo electrónico de la bandeja de entrada.	Alta	Alta
RFB-02	Cerrar aplicación	El sistema permitirá cerrar la aplicación.	Alta	Media
Categoría: Visualización				
RFB-03	Visualizar mensaje	El sistema permitirá visualizar los datos extraídos del mensaje: fecha, hora y mensaje original.	Alta	Alta
Categoría: Solicitud de verificación				
RFB-04	Enviar petición de verificación	El sistema permitirá enviar una petición de verificación del texto.	Alta	Alta
RFB-05	Notificar petición de verificación	El sistema debe notificar al usuario cada vez que se envía una petición.	Alta	Alta
Tipo: No funcional				
RNFB-01	Idioma	La aplicación únicamente mostrará la información en inglés.	Media	Alta
RNFB-02	Mensajes de error	El sistema debe proporcionar mensajes de error para informar al usuario.	Alta	Alta
RNFB-03	Zona horaria	El sistema permitirá mostrar la fecha y hora únicamente con zona horaria de España.	Alta	Alta
RNFB-04	Periodo de verificación	Los técnicos IT deberán verificar los correos en un plazo de 24 horas desde que se hizo la solicitud.	Alta	Alta
RNFB-05	Seguridad	Los usuarios deberán abrir un correo electrónico para poder usar el sistema.	Alta	Alta

TABLA 4.4. REQUISITOS SOFTWARE DE CA

Requisitos de Software de CA				
Tipo: Funcional				
Categoría: Inicio de Sesión				
Id	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RFC-01	Iniciar sesión	Los usuarios podrán iniciar sesión introduciendo: un id de trabajador y una contraseña.	Alta	Alta
RFC-02	Cancelar inicio de sesión	El sistema permitirá cancelar el inicio de sesión.	Alta	Media
RFC-03	Cerrar la sesión	El sistema permitirá cerrar la sesión.	Alta	Media
Categoría: Visualización				
RFC-04	Visualizar datos del usuario	El sistema permitirá visualizar el id del trabajador, la foto de perfil y el nombre del usuario una vez iniciada la sesión.	Alta	Baja
RFC-05	Visualizar lista empresas	El sistema permitirá visualizar una base de datos con una lista de empresas.	Alta	Media
RFC-06	Visualizar datos de la empresa	El sistema permitirá visualizar el correo y nombre de la empresa una vez seleccionada una empresa.	Alta	Baja
RFC-07	Visualizar base de datos "*Watermarks Database*"	El sistema permitirá visualizar la base de datos "*Watermarks Database*" con las marcas de agua almacenadas formadas por su id, kw y watermark.	Alta	Alta
RFC-08	Visualizar base de datos "*Requests Database*"	El sistema permitirá visualizar la base de datos "Requests Database" con las peticiones de verificación guardadas compuestas por su id y el texto original.	Alta	Alta
Categoría: Verificación				
RFC-09	Verificar petición	El sistema permitirá verificar cada petición.	Alta	Alta
Categoría: Algoritmo				
RFC-10	Comprobar el id	El sistema internamente permitirá comprobar que el id de la petición coincide con algún id de las bases de datos de las marcas de agua almacenadas.	Alta	Alta
RFC-11	Buscar palabra más repetida	El sistema internamente permitirá buscar la palabra más repetida en el texto.	Alta	Alta

RFC-12	Definir *kw* o clave del algoritmo	El sistema internamente podrá usar la palabra más repetida encontrada como *kw* o clave para el algoritmo.	Alta	Alta
RFC-13	Definir *watermark* del mensaje	El sistema internamente permitirá almacenar la frecuencia de las palabras anteriores y posteriores al *kw* en un watermark integrado en el mensaje junto al *kw*.	Alta	Alta
RFC-14	Comprobar el *kw*	El sistema internamente permitirá comprobar que los *kw* de la petición y del mensaje almacenado de id idéntico coinciden.	Alta	Alta
RFC-15	Comprobar el *watermark*	El sistema internamente permitirá comprobar que las marcas de agua de la petición y del mensaje almacenado de id idéntico coinciden.	Alta	Alta
Categoría: Notificación				
RFC-16	Notificar fases algoritmo	El sistema debe notificar al usuario cada vez que se realiza una fase del algoritmo.	Alta	Alta
Categoría: Envío				
RFC-17	Enviar mensaje con resultados	El sistema enviará un mensaje al emisor y/o al destinatario de la petición con el resultado de la verificación.	Alta	Alta
Categoría: Borrar base de datos				
RFC-18	Eliminar petición de la base de datos	El sistema deberá eliminar las peticiones de la base de datos "*Requests Database*" una vez verificadas.	Alta	Alta
RFC-19	Eliminar marca de agua de la base de datos	El sistema deberá eliminar las marcas de agua de la base de datos "*Watermarks Database*" una vez verificadas.	Alta	Alta
Tipo: No funcional				
RNFC-01	Idioma	La aplicación únicamente mostrará la información en inglés.	Media	Alta
RNFC-02	Mensajes de error	El sistema debe proporcionar mensajes de error para informar al usuario.	Alta	Alta
RNFC-03	Validar datos introducidos	El sistema debe validar los datos introducidos por el usuario.	Alta	Alta
RNFC-04	Zona horaria	El sistema permitirá mostrar la fecha y hora únicamente con zona horaria de	Alta	Alta

		España.		
RNFC-05	Periodo de verificación	Los técnicos IT deberán verificar los correos en un plazo de 24 horas desde que se hizo la solicitud.	Alta	Alta
RNFC-06	Seguridad	Los usuarios deberán iniciar sesión para poder usar el sistema.	Alta	Alta
RNFC-07	Seguridad	La contraseña debe almacenarse en hash.	Alta	Alta

4.4. Casos de Uso

Para definir el comportamiento del sistema y la interacción con otros usuarios o sistemas es necesario realizar los casos de uso. Al tener varias aplicaciones, esta herramienta puede ser muy útil para entender cada sistema individualmente y el conjunto global. Para ello, primero se muestran los diagramas de los casos de uso y después, se definen más en detalle.

4.4.1. Diagramas de casos de uso

Primero, hay que identificar al usuario que va a interactuar con el sistema. Al tratarse de tres aplicaciones, van a ser tres tipos de usuarios o actores: empleado de la empresa (fig. 4.4.), cliente (fig. 4.5.) y empleado TIC de CA (fig. 4.6.). Para definir correctamente las interacciones, es necesario realizar tres diagramas diferentes de casos de uso por cada tipo de usuario. Se han seleccionado las interacciones principales.

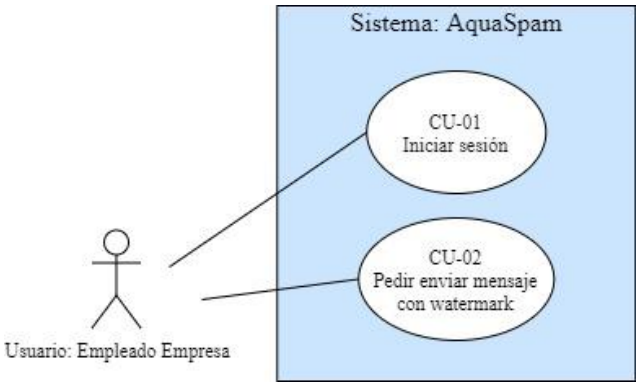


Fig. 4.4. Diagrama de casos de uso de AquaSpam

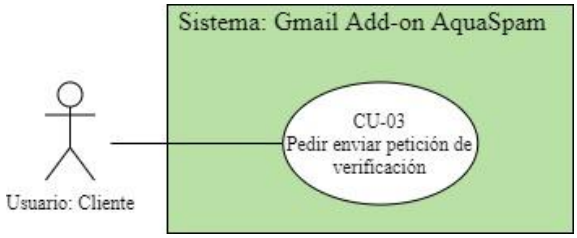


Fig. 4.5. Diagrama de casos de uso de Gmail Add-on AquaSpam

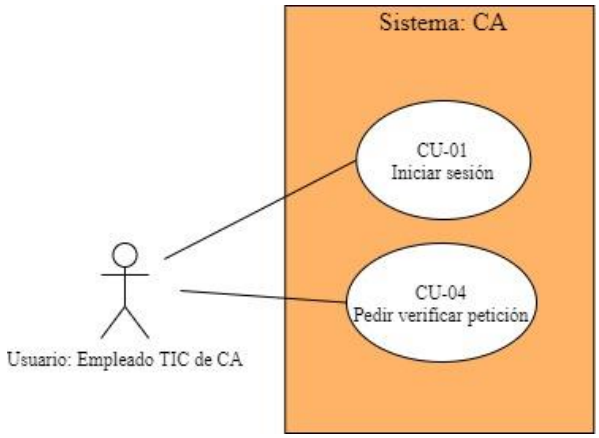


Fig. 4.6. Diagramas de casos de uso de CA

Como se puede observar, hay casos de uso que coinciden en más de un diagrama: CU-01 y CU-02. Pero se han realizado tres diagramas diferentes para una mejor comprensión del sistema completo.

4.4.2. Definición de casos de uso

En esta sección, se describen los casos de uso con más detalle siguiendo la plantilla de la tabla A.2. Todos los casos de uso definidos (tabla 4.5, tabla 4.6, tabla 4.7 y tabla 4.8) tienen un mismo autor, autora de este proyecto.

TABLA 4.5. DEFINICIÓN CASO DE USO CU-01

Id: CU-01
Nombre: Inicio de sesión
Descripción: Iniciar sesión
Actores: Empleado de la empresa y empleado TIC de CA
Precondiciones: • La aplicación está abierta. • El usuario está navegando en una ventana de inicio de sesión o *login*.
Postcondiciones: • El usuario se encontrará navegando en la siguiente ventana.
Flujo Normal: 1. El usuario debe introducir las credenciales y la contraseña adecuadas. 2. El usuario debe pulsar el botón "*Login*".

TABLA 4.6. DEFINICIÓN CASO DE USO CU-02

Id: CU-02
Nombre: Solicitud de envío de mensaje con watermark
Descripción: Pedir enviar mensaje con watermark
Actores: Empleado de la empresa.
Precondiciones: • La aplicación está abierta.

• El usuario está navegando en la ventana de creación de mensaje, posterior al inicio de sesión.
Postcondiciones: • El usuario se encontrará navegando en la misma ventana, pero aparecerá una ventana pequeña de información para notificar el envío.
Flujo Normal: 1. El usuario debe rellenar los campos: *To*, *Subject* y *Message*. En el campo "*To*", se debe escribir un correo válido. 2. El usuario debe pulsar el botón "*Send*".

TABLA 4.7. DEFINICIÓN CASO DE USO CU-03

Id: CU-03
Nombre: Solicitud de envío de petición de verificación
Descripción: Pedir enviar petición de verificación
Actores: Cliente
Precondiciones: • La aplicación está abierta. • El usuario está navegando por la ventana principal de la aplicación. • El usuario ha seleccionado previamente un correo electrónico de la bandeja de entrada.
Postcondiciones: • El usuario se encontrará navegando en la siguiente ventana. La ventana informativa de petición enviada.
Flujo Normal: 1. El usuario debe pulsar el botón "*Verify*".

TABLA 4.8. DEFINICIÓN CASO DE USO CU-04

Id: CU-04
Nombre: Solicitud de verificación de petición
Descripción: Pedir verificar petición
Actores: Empleado TIC de CA.
Precondiciones: • La aplicación está abierta. • El usuario está navegando en la ventana de la base de datos de peticiones.
Postcondiciones: • El usuario se encontrará navegando en la misma ventana, pero aparecerán ventanas informativas de las fases del algoritmo de verificación. • Los resultados se enviarán en forma de mensaje electrónico y la petición se borrará de la base de datos de peticiones. La marca de agua correspondiente se borrará de su base de datos solamente si el proceso de verificación se realiza con éxito.
Flujo Normal: 1. El usuario debe seleccionar una de las peticiones. 2. Una vez seleccionado, el usuario debe pulsar el botón "*Verify*".

5. DISEÑO

5.1. Introducción y proceso de diseño

Tras realizar el análisis de requisitos y de casos de uso, se diseña el sistema. Primero, se expone la idea principal y después, se muestra la arquitectura software del sistema.

El sistema está compuesto por tres aplicaciones, como se indica en la figura 5.1. El correo electrónico de la empresa se utiliza como **base de datos** para almacenar tanto las marcas de agua generadas como las peticiones de verificación. Las empresas utilizan **AquaSpam** para enviar correos electrónicos con un mensaje y su watermark a sus clientes. A su vez, esta marca de agua se almacena en el correo de la empresa junto al id, es decir, la fecha y la hora de creación. Se almacena bajo el asunto "*Watermak Created & Stored."* Los clientes utilizan **Gmail Add-on AquaSpam** para pedir la verificación. Se almacena la petición en el mismo correo de la empresa bajo el asunto "*Verification request for CA*". Los técnicos o trabajadores de AquaSpam hacen consultas de las bases de datos mediante la aplicación **CA**. Además, utilizan la aplicación para ejecutar la verificación. Automáticamente, se envían los resultados del algoritmo y finalmente, se borran de las bases de datos para ahorrar espacio.

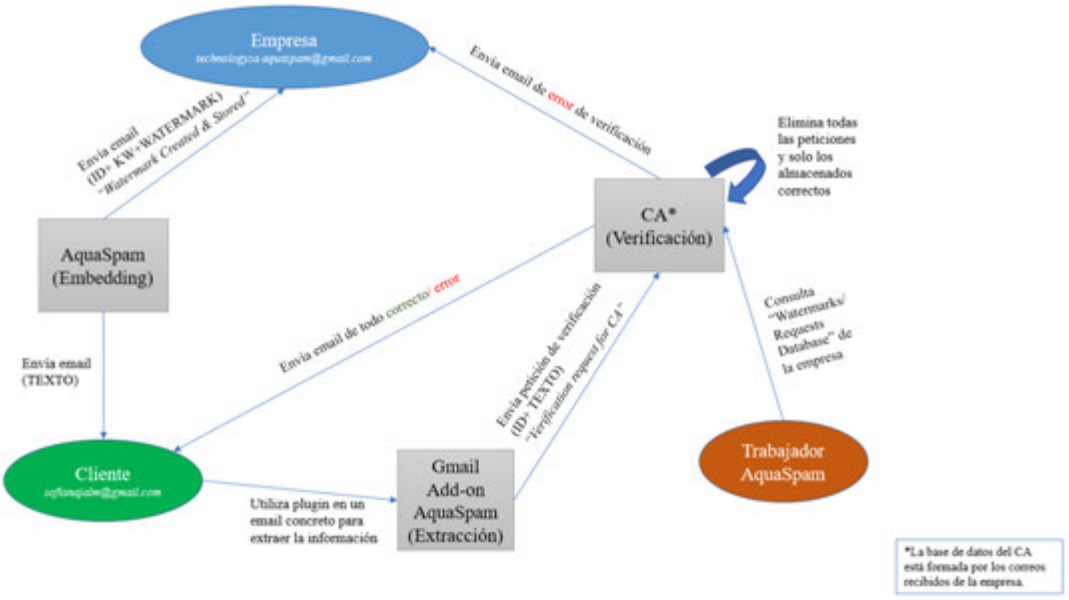


Fig. 5.1. Esquema idea principal del sistema

A continuación, se va a detallar la arquitectura definitiva del diseño software del sistema. En la figura 5.2., se pueden observar todos los componentes software que forman parte del sistema. Cada componente software se corresponde con una aplicación. El componente software llamado "*Gmail add-on AquaSpam*" necesita del servicio previo del componente software llamado "*AquaSpam*", es decir, que se haya enviado un email con el watermark al cliente. A su vez, para que el componte llamado

"*CA*" pueda verificar, necesita la petición previa del componente Gmail add-on AquaSpam.

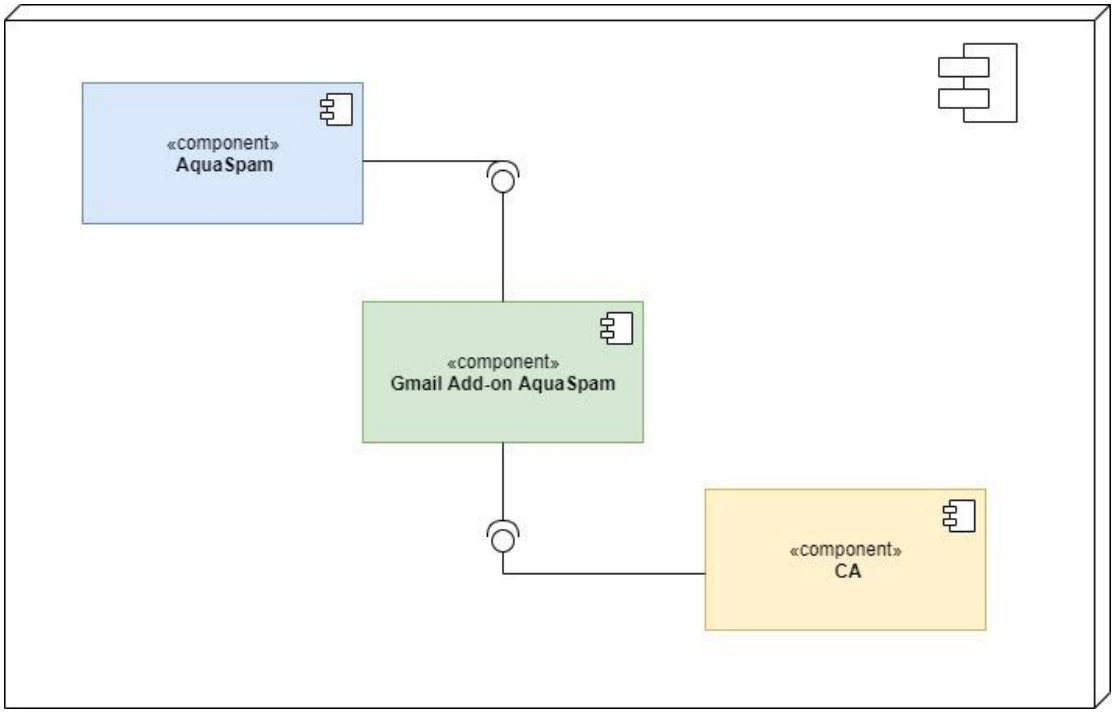


Fig. 5.2. Diagrama de componentes del sistema

5.2. Modelos de clase

En esta sección, se va a explicar detalladamente cada componente software y sus clases. Las clases, los atributos y los métodos que aparecen son los principales.

5.2.1. Componente AquaSpam

Primero, se debe iniciar sesión para poder utilizar esta aplicación. En este componente software, se puede enviar mensajes a través del correo electrónico con una marca de agua o *watermark* incrustada. Cada inicio de sesión permite crear uno o varios emails, incluso ninguno si no se quiere enviar ningún correo electrónico. El modelo de clases de este componente se representa en la figura 5.3.

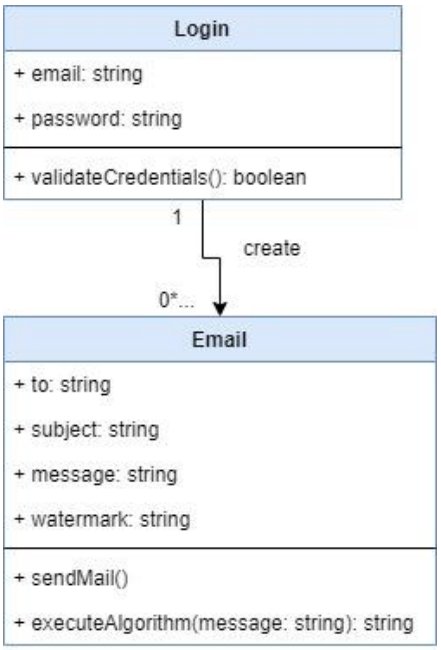


Fig. 5.3. Modelos de clase del componente AquaSpam

5.2.2. Componente Gmail Add-on AquaSpam

Primero, se debe seleccionar y abrir un correo electrónico de la bandeja de entrada. En este componente software, se puede extraer la información del mensaje recibido y enviar una petición de verificación con esa información extraída. El modelo de clases de este componente se representa en la figura 5.4.

Fig. 5.4. Modelos de clase del componente Gmail Add-on AquaSpam

5.2.3. Componente CA

Primero, se debe iniciar sesión para poder utilizarla. En este componente software, se puede consultar bases de datos y verificar peticiones, es decir, que coincida con alguna de las marcas de aguas almacenadas. Por cada inicio de sesión, se pueden consultar varias bases de datos debido a que los trabajadores pueden consultar distintas empresas desde su perfil. Pero en este proyecto, se enfoca en una sola empresa. Además, puede haber una o varias peticiones de las cuales se pueden verificar una única vez cada una. El modelo de clases de este componente se representa en la figura 5.5.

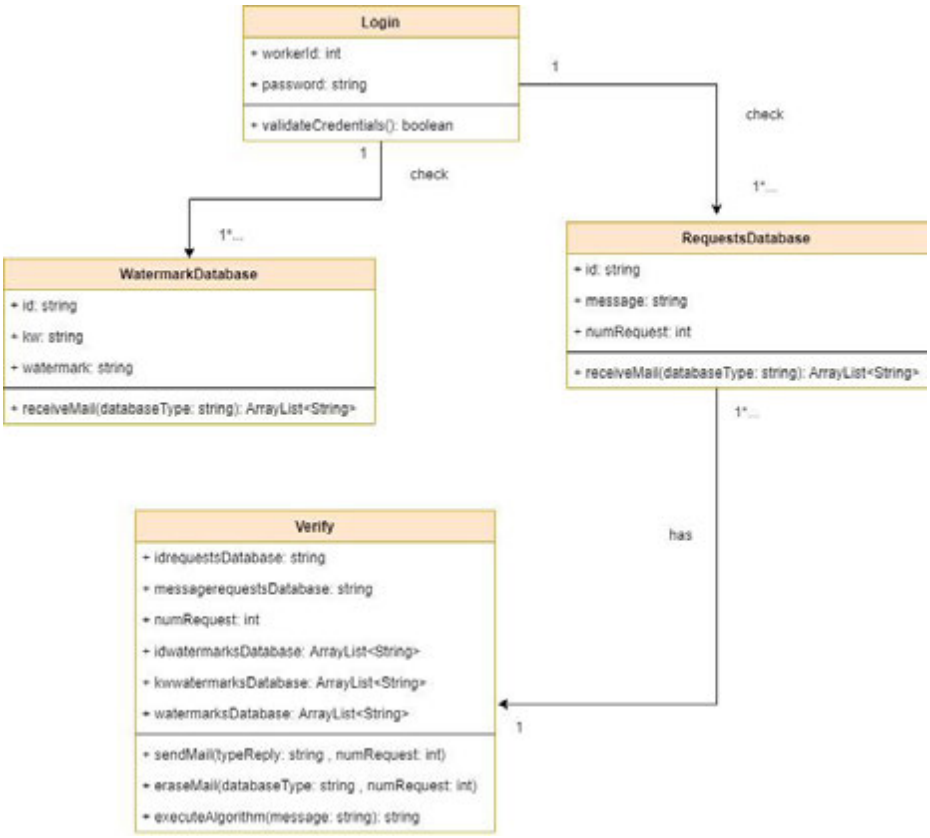


Fig. 5.5. Modelos de clase del componente CA

5.3. Diagramas de secuencia

A partir de cada caso de uso se extrae un diagrama de secuencia. Los diagramas de secuencia muestran la interacción y comportamiento dinámico entre usuario y objetos, es decir, a lo largo del tiempo. En los diagramas de secuencia solamente se representa el flujo normal, no los alternativos.

5.3.1. Inicio de sesión (CU-01)

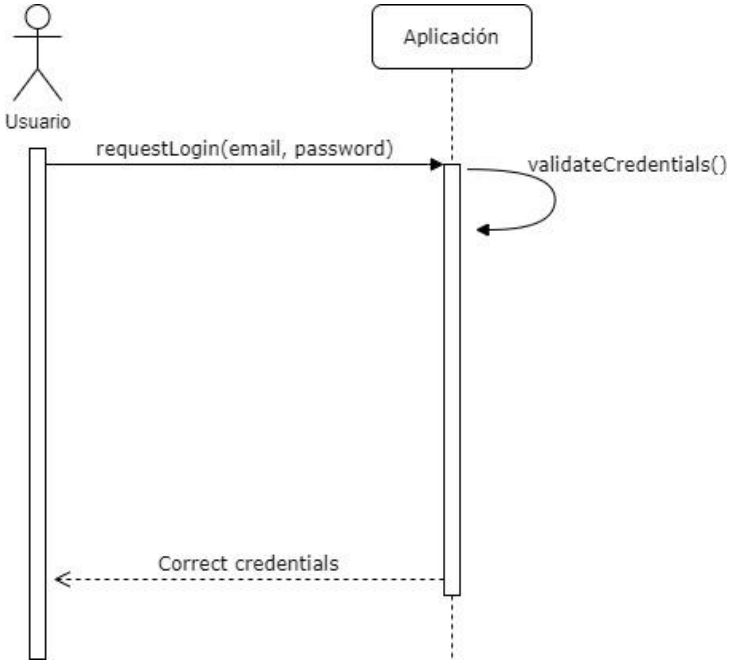


Fig. 5.6. Diagrama de secuencia CU-01 AquaSpam

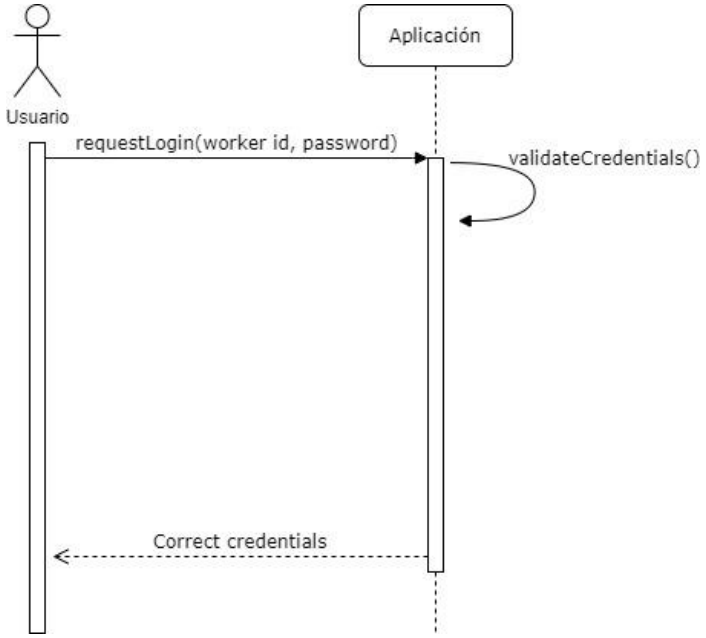


Fig. 5.7. Diagrama de secuencia CU-01 CA

En el caso de la app AquaSpam (fig. 5.6.), el usuario rellena los campos de su dirección de correo electrónico de Gmail y su contraseña. En el caso de la app CA (fig. 5.7.), el usuario rellena los campos de su id de trabajador y su contraseña. Al pulsar el botón "*Login*", la aplicación valida las credenciales introducidas.

5.3.2. Solicitud de envío de mensaje con watermark (CU-02)

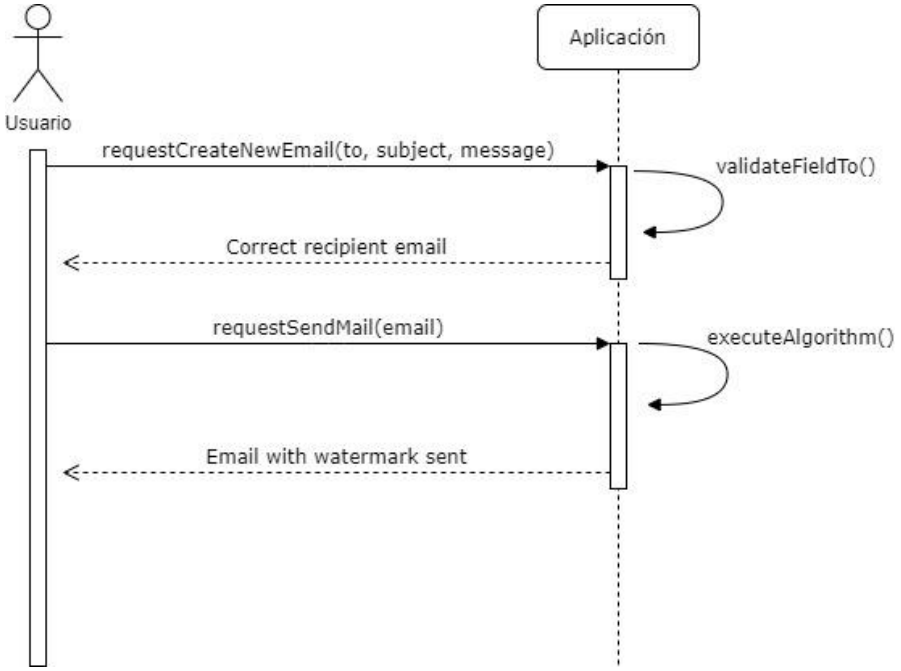


Fig. 5.8. Diagrama de secuencia CU-02

Para crear un nuevo mensaje (fig. 5.8.) es necesario que el usuario rellene los campos con la dirección de correo del destinatario (*to* o *recipient email*), el asunto (subject) y el mensaje. Al pulsar el botón "*Send*", la aplicación valida las credenciales introducidas y ejecuta el algoritmo.

5.3.3. Solicitud de envío de petición de verificación (CU-03)

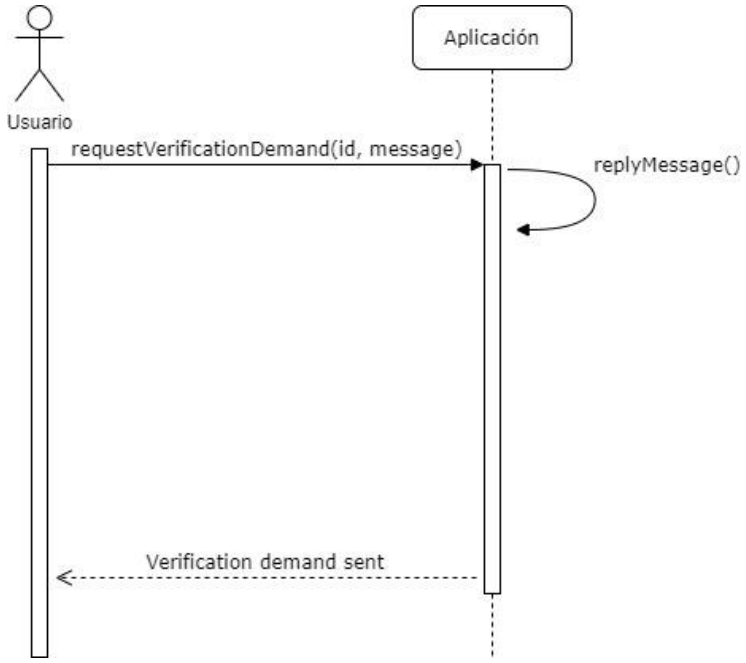


Fig. 5.9. Diagrama de secuencia CU-03

El usuario pulsa el botón "*Verify*". Se envía la solicitud de verificación (fig. 5.9.) junto a los elementos extraídos del mensaje: id y texto original.

5.3.4. Solicitud de verificación de petición (CU-04)

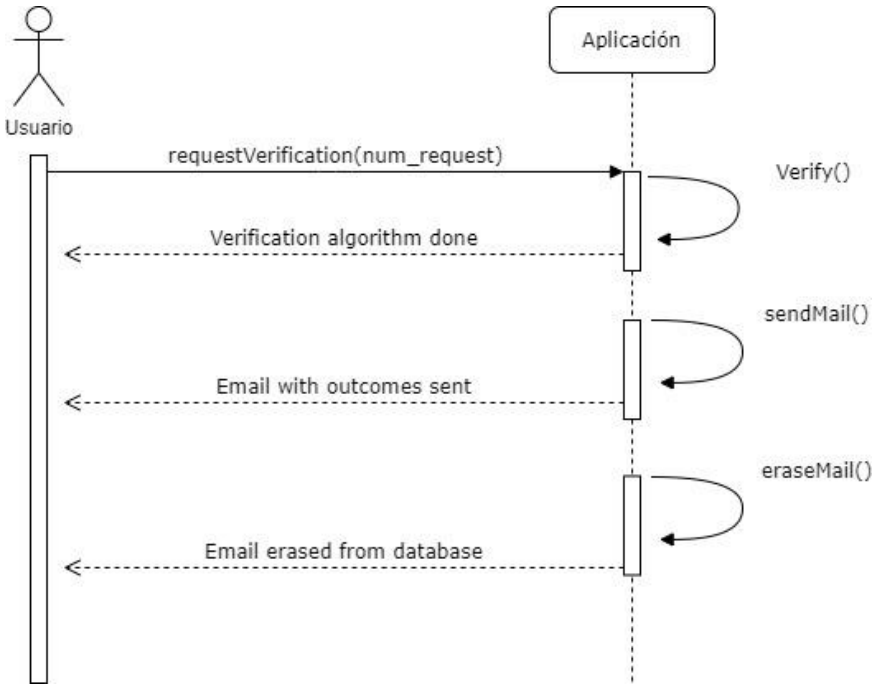


Fig. 5.10. Diagrama de secuencia CU-04

El usuario pulsa el botón "*Verify*". Se ejecuta el algoritmo de verificación de la clase *Verify.java*. Al finalizar, automáticamente, la aplicación envía los resultados por email y borra las peticiones y marcas de aguas de sus bases de datos correspondientes. El diagrama de secuencia del caso de uso cuarto, se representa en la figura 5.10.

5.4. Marco regulador

5.4.1. Análisis de la legislación aplicable

El estudio de los aspectos legales es fundamental en este proyecto, ya que, al ser aplicaciones software necesitan unas licencias de uso y condiciones y unas políticas. Es importante que las aplicaciones sigan la legislación española y europea para evitar cualquier tipo de sanción. Pero, sobre todo, para aumentar la confianza y seguridad de los usuarios. En las tres aplicaciones, se abarcarán tres legislaciones: términos y condiciones de uso, política de privacidad y política de cookies.

En primer lugar, toda aplicación software tiene unos **términos y condiciones de uso**. Esas condiciones las tendrán que leer los usuarios y aceptarlas. Todo aquello escrito y aceptado por el usuario evitará reclamos futuros. Si se hiciera un mal uso de una de las tres aplicaciones, unos términos y condiciones de uso bien redactados y transparentes permite eximirse de cualquier responsabilidad. En ese documento, se tendrán en cuenta cuestiones como la privacidad y la gestión de la información personal de los usuarios. Además, es necesario añadir los permisos para que estas aplicaciones

accedan a los correos electrónicos de Gmail [28]. Esta medida está regulada por el artículo 10 de la **Ley de Servicios de la Sociedad de la Información (L.S.S.I.)** [29]. Esta ley se enfoca en la cesión de datos de los usuarios. Pero como se tratan con bastantes datos personales y relevantes de los usuarios es necesario tener su propia política.

Los formularios de inicio de sesión donde hay que aportar el correo electrónico y la contraseña del correo de Gmail tienen que estar regulados por la **política de privacidad**. Al final, si esos datos no se gestionan de forma confidencial, estos se podrían ver comprometidos. Incluso, los datos almacenados en las bases de datos tienen que estar protegidos. En la política de privacidad, se tiene que pedir el consentimiento del usuario para acceder a sus datos personales desde la aplicación externa CA. Esa aplicación tiene que seguir estrictamente la **ley Orgánica de Protección de Datos (LOPD)** para almacenar y tratar esos datos personales. La **Agencia Española de Protección de Datos (AEPD)** se encarga del cumplimiento de esta ley y es donde hay que dar de alta esta política [30]. A su vez, toda Europa se rige por el **Reglamento General de Protección de datos (RGDP)** [31].

Los usuarios podrán borrar, aceptar o bloquear las cookies. En **la política de cookies**, se avisa al usuario del registro de los datos de los usuarios que utilicen la aplicación y de sus hábitos a la hora de navegar [32]. Se encuentra en el apartado segundo del artículo 22 de la Ley 34/2002, del 11 de julio, de *Servicios de la Sociedad de la Información y de Comercio Electrónico (LSSI-CE).* Además, la directiva *ePrivacy* de 2002 sobre privacidad, protección de datos y comunicaciones electrónicas [33] y el artículo 6 del RGDP [34] sobre la licitud del tratamiento de los datos del usuario, están fuertemente relacionadas con la política de cookies.

Cada vez se están creando nuevas regulaciones y/o actualizaciones de las ya existentes. Por ejemplo, el **RGDP** entró en vigor en 2016 pero no se empezó a aplicar hasta mayo del 2018 [35].

6. IMPLEMENTACIÓN

Primero, se expone el algoritmo utilizado y después, se explica brevemente lo esencial de las tres aplicaciones. Para verlo en práctica, en el apartado *Anexo: Manual de Usuario* se adjunta una pequeña guía a través de las tres aplicaciones.

6.1. Algoritmo

El algoritmo utilizado se llama "*Zero-Watermarking*" [13]. Consiste en dado un texto plano y la selección de una clave o *keyword* (*kw*), se genera una marca de agua o watermark para identificar la autoría del archivo al que se le aplica el algoritmo, en este caso quién envía el mensaje.

Para la selección de la *keyword*, el algoritmo escoge la palabra con mayor frecuencia del texto. Para crear la marca de agua se buscan todas las palabras en el texto que coincidan con la clave escogida en orden. Se contabilizan las letras de las palabras inmediatamente anterior y posterior de la clave encontrada. Por tanto, la marca de agua tiene la palabra clave y una secuencia de números. Este primer proceso se llama "*Embedding*". En la figura 6.1., se muestra a modo visual como se genera una marca de agua.

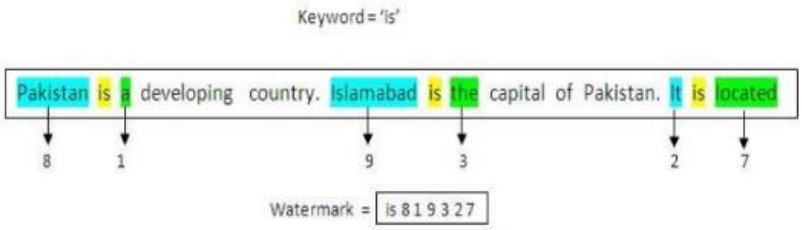


Fig. 6.1. Generación de la marca de agua [13]

Después, una entidad externa llamada *Certifying Authority* o *Certificate Authority* (*CA*), almacena la marca de agua junto a un id. El id utilizado es la fecha y hora en formato "*DD/MM/YY HH:MM: SS"*.

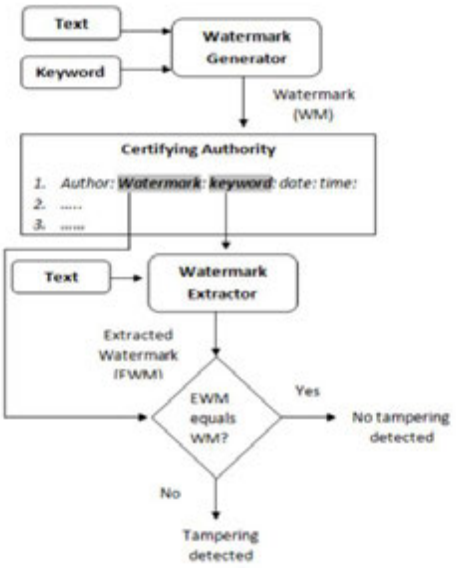


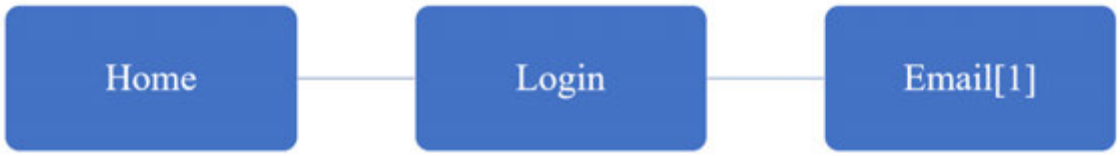
Fig. 6.2. Esquema del algoritmo Zero-Watermarking [13]

El último proceso es el de *extracción*. Es idéntico al de *embedding,* pero se hace sobre el mensaje recibido no sobre el enviado. Se realiza para comprobar que el mensaje no se ha manipulado y que el autor del mensaje se ha verificado correctamente. Finalmente, se accede al almacenamiento de CA para comprobar que las marcas de agua coinciden. En la figura 6.2., se puede observar el esquema general del algoritmo.

6.2. Aplicaciones

Tras exponer el algoritmo, se expone la implementación de cada una de las aplicaciones propuestas.

Primero, la empresa decide enviar un correo electrónico a través de la aplicación AquaSpam. Entonces, se ejecuta el algoritmo "*Embedding*" y posteriormente se envía el mensaje con la marca de agua correspondiente. En la figura 6.3., se muestran las pantallas de la aplicación AquaSpam.



(Este app lo utiliza la empresa)
[1] Se ejecuta el algoritmo de **Embedding** utilizando el texto como entrada. En la clase *Main*, se compilan los pasos por orden: cálculo del kw (palabras más frecuente) y cálculo de watermark a partir de las longitudes de las palabras anteriores y posteriores cada vez que aparece el kw. Se envía un mensaje desde el email de la empresa al cliente correspondiente. A su vez, se envía un mensaje a si mismo para almacenar el **id**, **kw** y **watermark**. El email se usa como base de datos. Los mensajes de almacenamiento tienen el asunto "*Watermark Created & Stored*".

Fig. 6.3. Esquema app AquaSpam

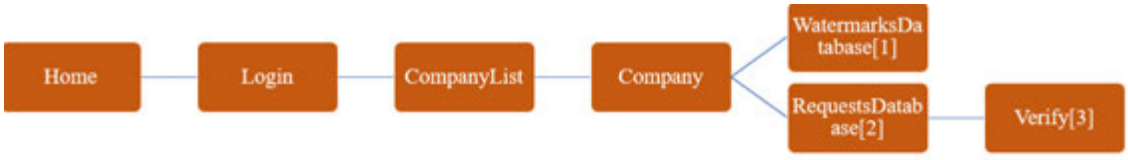
Segundo, el cliente hace una solicitud de verificación del correo enviado por la empresa. Hace la petición a través del correo de Gmail utilizando Gmail add-on de

AquaSpam. En la figura 6.4., se muestran las pantallas de la aplicación Gmail add-on de AquaSpam.

(Este add-on lo utiliza el cliente)
[1] Muestra la extracción de información del mensaje de correo en Gmail. Los datos son: **la fecha y hora (id) y el texto**. Hay un botón "Verify" para pasar a la siguiente pantalla.
[2] Se responde al mensaje recibido (*reply*) con asunto "*Verification request for CA*" y como cuerpo del mensaje la información extraída previamente. A su vez, se muestra una ventana informando de que la petición ha sido enviada con éxito.

Fig. 6.4. Esquema app Gmail Add-on AquaSpam

Por último, la aplicación CA accede al correo de la empresa creado por y para AquaSpam (*technologysa.aquaspam@gmail.com*) y así gestionar todas las empresas de manera individualizada (simplificación del prototipo). En ese correo, se encuentran dos bases de datos, una almacena las marcas de agua (*WatermarkDatabase*) y la otra almacena las peticiones (*RequestsDatabase*). Los empleados de AquaSpam se encargarían de verificar esas peticiones. El algoritmo de verificación sería el llamado anteriormente "*extracción*". Este proceso se puede automatizar, es decir, se podría prescindir de los empleados, pero el desarrollo propuesto es manual para ver en detalle todos los procesos. En la figura 6.5., se muestran las pantallas de la aplicación CA.



(Este app lo utilizan los trabajadores técnicos de AquaSpam)
[1] Busco en el email de la empresa los mensajes con los **watermarks almacenados** (clase *ReceiveMail*) y extraigo el **id**, **kW** y **texto** para mostrarlos mejor.(Similar a una base de datos) Funciones: Mostrar.
[2] Busco en el email de la empresa los **mensajes de petición** (clase *ReceiveMail*) para verificar que es idéntico al mensaje que ha enviado la empresa. Extraigo el **id** y el **texto** para mostrarlos mejor. Funciones: Mostrar + Verificar.
[3] Primero, se comprueba que el **id** existe en la base de datos de los **watermarks almacenados** *(Step 1)*. Una vez verificado, se **recalcula** el **watermark** y **kw** con el mismo algoritmo usado para *AquaSpam* (clases *Embedding, Ocurrence y Main*) *(Step 2)* . Contrasto los resultados con los valores almacenados y se envían mensajes específicos a los clientes y a la empresa según el caso. Si es correcto se borran tanto la petición como el almacenado y si es incorrecto, solamente se borra la petición *(Step 3)*.

Fig. 6.5. Esquema app CA

Para iniciar sesión en las aplicaciones AquaSpam y CA es necesario usar una contraseña. Estas contraseñas tendrían que estar almacenadas en hash en las bases de datos para evitar robo de contraseñas si las bases de datos se vieran comprometidas. Esta funcionalidad no se ha llevado a cabo debido a que se trata de un prototipo y había elementos de mayor relevancia. Pero esto no quiere decir que no se haya tenido en cuenta.

7. EVALUACIÓN

En esta sección, se realizan una serie de ensayos para comprobar que cada una de las tres aplicaciones funciona correctamente. Sirven para un mejor entendimiento del sistema. Además, evalúan los requisitos previamente definidos.

7.1. Diseño del plan de pruebas de aceptación

Siguiendo la plantilla de la tabla A.3, se definen las pruebas de aceptación (tabla 7.1) en tres grupos, ya que, en este proyecto hay tres aplicaciones.

TABLA 7.1. PLAN DE LAS PRUEBAS DE ACEPTACIÓN

Pruebas de aceptación			
Aplicación: AquaSpam			
Id	**Requisitos**	**Entrada**	**Salida**
PAA-01	RFA-01, RNFA-03, RNFA-04	Se está navegando por la ventana de login o inicio de sesión y se escribe una dirección de correo y una contraseña y se pulsa el botón de "*Login*".	Se navega correctamente a la siguiente ventana de creación de e-mail.
PAA-02	RFA-02	Se está navegando por la ventana de login o inicio de sesión y se decide cancelar el inicio de sesión.	Se vuelve a la pantalla inicial de bienvenida.
PAA-03	RFA-03, RFA-06, RFA-07, RFA-08, RFA-09, RNFA-03	Se está navegando por la ventana de creación de email y se escribe una dirección de correo, un asunto y un mensaje. Se pulsa el botón de "*Send*".	Se realiza el algoritmo de watermarking con el texto como entrada. Después, se envía correctamente el mensaje con el watermark incluido. Por último, se notifica el envío.
PAA-04	RFA-04, RFA-05	Se está navegando por la ventana de creación de email, se escribe una dirección de correo, un asunto y un mensaje y se pulsa el botón "*Clear all fields*" y después, se pulsa el botón de "*Cancel*".	Se borrarían todos los campos y se volvería a la ventana anterior, la de inicio de sesión.
PAA-05	RFA-01, RNFA-02	Se está navegando por la ventana de login o inicio de sesión y se escribe una dirección de correo incorrecta y/o una contraseña incorrecta y se pulsa el botón de "*Login*".	Saldría un mensaje de error "*Error: Invalid Login*" y pediría introducir otra vez las credenciales correctas.
Aplicación: Gmail Add-on AquaSpam			
Id	**Requisitos**	**Entrada**	**Salida**
PAB-01	RFB-01, RNFB-05	Se está navegando por la ventana de la bandeja de entrada del correo electrónico y se selecciona un mensaje. Después, se pulsa el icono de la	Se abre la aplicación.

		aplicación en la parte inferior derecha.	
PAB-02	RFB-03, RFB-04, RFB-05, RNFB-03, RNFB-04	Se está navegando por la ventana del plugin y se pulsa el botón "*Verify*".	Se muestra una ventana con la información extraída del mensaje seleccionado. La información consiste en la fecha, hora y el mensaje original. Se envía una petición de verificación con los datos extraídos y se muestra una ventana informativa dando un plazo de 24 horas.
PAB-03	RFB-02, RFB-05, RNFB-04	Se está navegando por la ventana informativa de petición enviada y se pulsa el botón de cerrar.	Se cierra la aplicación, pero el contador de 24 horas sigue encendido, además, se guarda en la aplicación cuando se ha realizado petición. Porque cada vez que se abre, aparece la misma ventana informativa.
Aplicación: CA			
Id	**Requisitos**	**Entrada**	**Salida**
PAC-01	RFC-01, RNFC-03, RNFC-06	Se está navegando por la ventana de login o inicio de sesión y se escribe un id de trabajador y una contraseña y se pulsa el botón de "*Login*".	Se navega correctamente a la siguiente ventana de lista de empresas
PAC-02	RFC-04, RFC-05	Se está navegando por la lista de empresas y se pulsa una de ellas.	Se navega correctamente a la siguiente ventana de la empresa seleccionada.
PAC-03	RFC-03, RFC-04, RFC-05	Se está navegando por la ventana con la lista de empresas y se pulsa el botón de la esquina superior derecha.	Se cierra la sesión y se vuelve a la ventana anterior de inicio de sesión.
PAC-04	RFC-06, RFC-07, RNFC-04	Se está navegando por la ventana de la empresa seleccionada y se pulsa el botón "*Watermarks Database*".	Se abre la base de datos de las marcas de agua y se muestran sus id, kw y watermark.
PAC-05	RFC-06, RFC-08, RNFC-04	Se está navegando por la ventana de la empresa seleccionada y se pulsa el botón "*Requests Database*".	Se abre la base de datos de las peticiones y se muestran sus id y el texto original.
PAC-06	RFC-09, RFC-10, RFC-11, RFC-12, RFC-13, RFC-14, RFC-15,	Se está navegando por la ventana de las peticiones. Se pulsa el botón "*Verify*".	Comienza a realizarse el algoritmo de verificación. Se realizan con éxito las tres fases de comprobación: id, kw y watermark. A su vez, se notifica cada fase. Al terminar, se envía un mensaje al emisor y/o destinatario con los resultados. Por

	RFC-16, RFC-17, RFC-18, RFC-19		último, se elimina tanto la petición como la marca de agua de su base de datos correspondiente.
PAC-07	RFC-10, RFC-16, RFC-17, RFC-18, RNFC-02	Se está navegando por la ventana de las peticiones. Se pulsa el botón "*Verify*".	Comienza a realizarse el algoritmo de verificación. Falla el algoritmo en la primera fase: comprobación de id. Este es el caso de autenticación incorrecta. Se muestra un mensaje de error y se envía al cliente. El mensaje de asunto "*FAILED: Verification request for CA*" tiene la siguiente información: "*There is a high risk in this message. Id Authentication failed! Fraud detected. Erasing dangerous messages and introducing fraud sender in CA blacklist*". Además, se borra de la base de datos de las peticiones.

7.2. Resultados de las pruebas de aceptación

En este apartado, se muestran los resultados obtenidos (tabla 7.2) de las pruebas de aceptación definidas en la sección 7.1.

TABLA 7.2. RESULTADOS PRUEBAS DE ACEPTACIÓN

Pruebas de aceptación		
Aplicación: AquaSpam		
Id	**Requisitos**	**Resultado**
PAA-01	RFA-01, RNFA-03, RNFA-04	Superada
PAA-02	RFA-02	Superada
PAA-03	RFA-03, RFA-06, RFA-07, RFA-08, RFA-09, RNFA-03	Superada
PAA-04	RFA-04, RFA-05	Superada
PAA-05	RFA-01, RNFA-02	Superada
Aplicación: Gmail Add-on AquaSpam		
Id	**Requisitos**	**Resultado**
PAB-01	RFB-01, RNFB-05	Superada
PAB-02	RFB-03, RFB-04, RFB-05, RNFB-03, RNFB-04	Superada
PAB-03	RFB-02, RFB-05, RNFB-04	Superada
Aplicación: CA		
Id	**Requisitos**	**Resultado**

PAC-01	RFC-01, RNFC-03, RNFC-06	Superada
PAC-02	RFC-04, RFC-05	Superada
PAC-03	RFC-03, RFC-04, RFC-05	Superada
PAC-04	RFC-06, RFC-07, RNFC-04	Superada
PAC-05	RFC-06, RFC-08, RNFC-04	Superada
PAC-06	RFC-09, RFC-10, RFC-11, RFC-12, RFC-13, RFC-14, RFC-15, RFC-16, RFC-17, RFC-18, RFC-19	Superada
PAC-07	RFC-10, RFC-16, RFC-17, RFC-18, RNFC-02	Superada

Como se puede comprobar, se ha hecho un buen análisis y diseño, ya que, los resultados demuestran que se han **superado** todas las pruebas de aceptación.

8. GESTIÓN DEL PROYECTO

8.1. Planificación

En esta sección se muestra la organización de tareas llevada a cabo para la realización del proyecto. Finalmente, se concluye la duración total en días.

El desarrollo del código presentado en la tabla 8.3, muestra los días en los que se ha realizado este proyecto. Dada la situación que hemos atravesado por el Covid-19, los plazos han sufrido pequeñas variaciones con lo establecido en un primer momento, presentado en la tabla 8.1.

Planificación inicial

TABLA 8.1. PLANIFICACIÓN INICIAL DEL PROYECTO

Tarea	Días reales	Fecha de inicio	Fecha fin	Duración
Investigación	-3 días (julio) -9 días (agosto) -8 días (septiembre)	23/07/2019	26/09/2019	**20 días**
Desarrollo del código y Memoria: Implementación	-5 días (noviembre) -7 días (febrero) -10 días (marzo)	16/11/2019	10/03/2020	**22 días**
Introducción	-1 día (marzo)	11/03/2020	11/03/2020	**1 día**
Gestión de proyecto	-2 días (marzo)	12/03/2020	13/03/2020	**2 días**
Anexo: Manual de uso	-2 días (marzo)	16/03/2020	17/03/2020	**2 días**
Conocimientos previos	-1 día (marzo)	18/03/2020	18/03/2020	**1 día**
Análisis	-3 días (marzo)	19/03/2020	21/03/2020	**3 días**
Diseño	-2 días (marzo)	23/03/2020	24/03/2020	**2 días**
Evaluación	-1 día (marzo)	25/03/2020	25/03/2020	**1 día**
Estado del arte	-2 días (abril)	02/04/2020	03/04/2020	**2 días**
Documentación restante	-1 día (abril)	04/04/2020	04/04/2020	**1 día**
Total TFG	----------	23 de julio de 2019	4 de abril de 2020	**57 días**

Para una mejor visualización, se muestra esa planificación inicial de tareas mediante la herramienta del diagrama de Gantt (fig. 8.1.).

TABLA 8.2. PLANIFICACIÓN INICIAL DEL DIAGRAMA DE GANTT

TAREAS	INICIO	FINALIZACIÓN	DÍAS
Investigación	23-Jul	26-Sep	66
Implementación	16-Nov	10-Mar	116
Introducción	11-Mar	11-Mar	1
Gestión del proyecto	12-Mar	13-Mar	2
Anexo: Manual de uso	16-Mar	17-Mar	2
Conocimientos previos	18-Mar	18-Mar	1
Análisis	19-Mar	21-Mar	3
Diseño	23-Mar	24-Mar	2
Evaluación	25-Mar	25-Mar	1
Estado del arte	2-Apr	3-Apr	2
Documentación restante	4-Apr	4-Apr	1

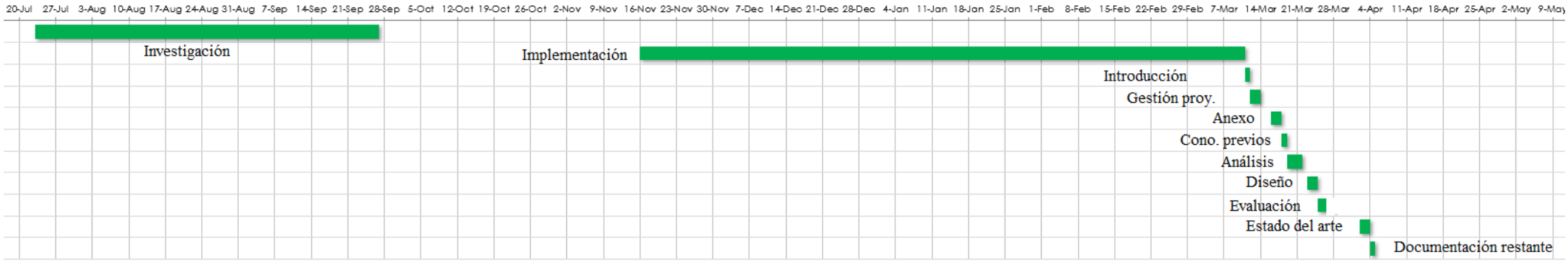


Fig. 8.1. Diagrama inicial Gantt

Planificación final

TABLA 8.3. PLANIFICACIÓN FINAL DEL PROYECTO

Tarea	Días reales	Fecha de inicio	Fecha fin	Duración
Investigación	-3 días (julio) -9 días (agosto) -8 días (septiembre)	23/07/2019	26/09/2019	**20 días**
Desarrollo del código y Memoria: Implementación	-5 días (noviembre) -7 días (febrero) -7 días (marzo) -9 días (abril) -1 día (mayo)	16/11/2019	02/05/2020	**29 días**
Introducción	-2 días (junio)	01/06/2020	02/06/2020	**2 días**
Gestión de proyecto	-3 días (junio)	3/06/2020	17/06/2020	**3 días**
Anexo: Manual de uso	-2 días (junio)	5/06/2020	6/06/2020	**2 días**
Conocimientos previos	-1 día (junio)	7/06/2020	7/06/2020	**1 día**
Análisis	-4 días (junio)	7/06/2020	10/06/2020	**4 días**
Diseño	-3 días (junio)	11/06/2020	13/06/2020	**3 días**
Evaluación	-1 día (junio)	14/06/2020	14/06/2020	**1 día**
Estado del arte	-3 días (junio)	15/06/2020	17/06/2020	**3 días**
Documentación restante	-1 día (junio)	17/06/2020	17/06/2020	**1 día**
Total TFG	----------	23 de julio de 2019	17 de junio de 2020	**69 días**

Para una mejor visualización, se muestra esa planificación final de tareas mediante la herramienta del diagrama de Gantt (fig. 8.2.).

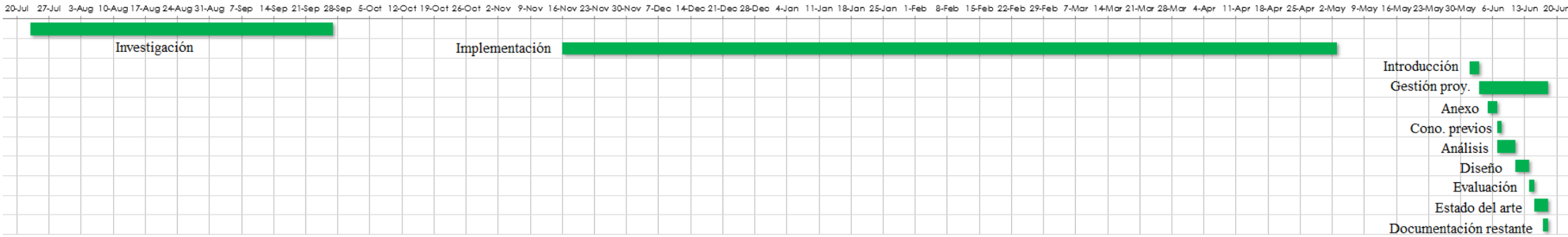


Fig. 8.2. Diagrama final de Gantt

TABLA 8.4. PLANIFICACIÓN FINAL DEL DIAGRAMA DE GANTT

TAREAS	INICIO	FINALIZACIÓN	DÍAS
Investigación	23-Jul	26-Sep	66
Implementación	16-Nov	2-May	169
Introducción	1-Jun	2-Jun	2
Gestión del proyecto	3-Jun	17-Jun	15
Anexo: Manual de uso	5-Jun	6-Jun	2
Conocimientos previos	7-Jun	7-Jun	1
Análisis	7-Jun	10-Jun	4
Diseño	11-Jun	13-Jun	3
Evaluación	14-Jun	14-Jun	1
Documentación restante	17-Jun	17-Jun	1

Conclusiones de la planificación

Tras observar las tablas 8.1 y 8.3, la planificación inicial era de 57 días reales terminando el 4 de abril y la planificación final fue de 69 días reales terminando el 17 de junio. Con esto se demuestra un retraso de **12 días reales**. Al observar los diagramas de Gantt (fig. 32, fig. 33, tabla 8.2 y tabla 8.4), se calculan 197 días en la planificación inicial y 264 días en la planificación final, es decir, **67 días de plazo** más en la planificación final. Por tanto, se demuestra que se ha tardado mucho más tiempo que lo planificado inicialmente.

8.2. Entorno socioeconómico

8.2.1. Presupuesto

Para realizar la estimación de costes es importante elegir una metodología específica. La usada para este proyecto se calcula en base a costes directos e indirectos. Los costes directos corresponden con los recursos humanos y materiales, es decir, gastos de personal, gastos de equipos, gastos software, gastos consumibles y gastos de viajes y dietas. Los costes indirectos corresponden con los gastos de electricidad e internet.

Debido a la situación del *COVID-19* hay muchos recursos que se han recortado como es el gasto de gasolina. En el periodo de julio a marzo, había clases presenciales en la Universidad Carlos III de Madrid por lo que a veces me quedaba en la biblioteca o la sala informática trabajando. En los meses restantes, el gasto del transporte desaparece.

Gastos directos

Gastos de personal

Solo ha habido una persona como mano de obra. He tenido que hacer de todos los roles típicos de cualquier proyecto de software, es decir, jefe de proyecto, analista, pruebas y programador. En la tabla 8.5, se considera el sumatorio de los sueldos actuales [36] de todos los roles descritos. Este coste al día incluye el coste de la seguridad social.

El proyecto se comenzó el 23 de julio de 2019 y terminó el 17 de junio de 2020, de los cuales en total se trabajaron 69 días. El jefe de proyecto ha estado a lo largo de todo el proyecto supervisando los demás roles.

TABLA 8.5. GASTOS DE PERSONAL

Cargo	Coste (€/día)	Días dedicados	Total (€)
Jefa de proyectos	175€/día	69	12.075 €
Ingeniera software	114,58€/día	30	3.437,40€
Analista de sistemas	133,33€/día	6	799,98€
Ingeniera de pruebas	118,75€/día	4	475€
Programadora	87,50€/día	29	2.537,50€
		Coste imputable total	**19.324,88€**

Gastos de equipos

A lo largo del proyecto se han utilizado los siguientes dispositivos:

- Pc sobremesa ideacentre 510S-08IKL, procesador i5-7400, 12 GB de RAM y 3 GHz. Incluye ratón y teclados con cable.
- Portátil HP 15-dw2004ns, procesador Intel Core i5-1035G1, 8GB de RAM y 1.19 GHz. Incluye ratón inalámbrico externo.

Para calcular los costes correctamente hay que tener en cuenta varios valores: el coste del equipo, los meses dedicados, el periodo de depreciación y el porcentaje de uso dedicado. Todo el proyecto ha tenido una duración de 9 meses excluyendo vacaciones y meses de exámenes. En la tabla 8.6, se exponen los gastos de los equipos calculados con la siguiente fórmula:

$$F = \frac{C}{P} * D$$

Siendo:

- F: Coste final imputable sin IVA.
- C: Coste del equipo sin IVA.
- D: Número de meses empleados en el proyecto.

- P: Periodo de depreciación en meses.
- El porcentaje de uso dedicado es del 100%, se multiplicaría el resultado por 1.

TABLA 8.6. GASTOS DE EQUIPOS

Descripción	Coste (€)	Dedicación (meses)	Periodo depreciación (meses)	Coste imputable (€)
Pc sobremesa	799,00€	9 meses	36 meses	199,75€
Pc portátil	800,00€	9 meses	36 meses	200,00€
			Coste imputable total	**399,75€**

Gastos software

Para los gastos del software en la tabla 8.7, se utiliza la misma fórmula de coste imputable anterior. Los costes son sin IVA y de la tienda oficial de Microsoft.

También, se ha utilizado *Eclipse Photon* como herramienta software para desarrollar y ejecutar el código de lenguaje JAVA. Pero es de código abierto, sin coste ni gasto.

TABLA 8.7. GASTOS SOFTWARE

Descripción	Coste (€)	Dedicación (meses)	Periodo depreciación (meses)	Coste imputable (€)
Windows 10 Home	290,00€	9 meses	36 meses	72,50€
Microsoft Office 365 Empresa Premium	33,80€	9 meses	36 meses	8,45€
GSuite Business*	10,40€	9 meses	36 meses	2,6€
			Coste imputable total	**83,55€**

**GSuite Business se ha utilizado para desarrollar el add-on de Gmail.*

Gastos consumibles

Los únicos gastos consumibles han sido los recursos materiales de oficina como son bolígrafos, lápices, folios, cuadernos… mostrados en la tabla 8.8.

TABLA 8.8. GASTOS CONSUMIBLES

Descripción	Coste (€)
Material de Oficina	**26,00€**

Gastos de viajes y dietas

Como se ha comentado antes estos gastos serían mayores si no hubiera ocurrido la pandemia global pero como empezó en el segundo cuatrimestre, hay costes, pero reducidos. El abono de transporte se usó 4 meses y de esos 4 meses me quedaba en la biblioteca únicamente los viernes. Es decir, 1 menú de la universidad por semana. Por tanto, fueron 16 comidas. Cada bono de comedor de 10 comidas cuesta 52,00€ y el menú normal cuesta 5,60€. Por ello, 1 bono y 6 comidas individuales son 85,60€. En la tabla 8.9 se pueden observar estos gastos de viajes y dietas.

TABLA 8.9. GASTOS VIAJES Y DIETAS

Descripción	Coste (€)
Abono transporte	80,00€
Comidas	85,60€
Total	**165,60€**

Ahora, se suman todos los gastos directos en la tabla 8.10.

TABLA 8.10. GASTOS DIRECTOS

Descripción	Coste (€)
Gastos de personal	19.324,88 €
Gastos de equipos	399,75€
Gastos software	83,55€
Gastos consumibles	26,00€
Gastos de viajes y dietas	165,60€
Total	**19.999,78€**

Gastos indirectos

Debido al *Covid-19*, la mayoría del proyecto se ha llevado a cabo en casa, por ello, el gasto mayoritario es de la electricidad y del internet. En la tabla 8.11, se suman todos los gastos indirectos.

TABLA 8.11. GASTOS INDIRECTOS

Descripción	Coste (€/mes)	Uso (meses)	Total (€)
Electricidad	32,16€	9 meses	289,44€
Internet	33,00€	9 meses	297,00€
		Coste total	**586,44€**

Presupuesto final

TABLA 8.12. PRESUPUESTO FINAL

Descripción	Coste (€)
Gastos directos (sin IVA)	19.999,78€
Gastos indirectos (sin IVA)	586,44€
Total (sin IVA)	20.586,22€
Riesgo (15%)	3.087,93€
Total (sin IVA con riesgo)	23.674,15€
Beneficios (15%)	3.551,12€
Total (sin IVA con riesgo y beneficio)	27.225,27 €
IVA (21 %)	5.717,31€
Total (con IVA: 21%, riesgo y beneficio)	**32.942,58 €**

El presupuesto total de este proyecto calculado en la tabla 8.12 es de **TREINTA Y DOS MIL NOVECIENTAS CUARENTA Y DOS CON CINCUENTA Y OCHO EUROS**.

8.2.2. Impacto socioeconómico

En esta sección se va a realizar un análisis del impacto del proyecto.

Con respecto al impacto social, esta aplicación de watermarking para la detección de spam ayudará a los clientes de numerosas empresas a identificar y evitar los correos fraudulentos. Gracias a la aplicación se velará por la seguridad de ambos: empresa y cliente. Además, se reducirá la criminalidad cibernética y como consecuencia, aumentará la prevención y concienciación por parte de los usuarios.

Aumentar la seguridad ahorra muchísimo dinero a las empresas e incrementa la probabilidad de nuevos clientes. Por ello, es importante mencionar el peligro de las fugas de información o robos de datos de los clientes y de la empresa. Cada vez que ocurre se pierde bastante dinero. Este proyecto requerirá de una plantilla pequeña de personal para mantener las aplicaciones (*AquaSpam, Add-on y CA*), servidores y tal vez, una oficina donde llevar todo a cabo. No solo se ahorra dinero si no tiempo. Al final es un proceso automático y rápido. La empresa envía un mensaje, el cliente pide

verificación, el CA verifica que todo está correcto y se envían los correspondientes mensajes informativos con los resultados de la verificación, es decir, si se ha verificado correctamente o ha habido algún tipo de error.

El uso de medios digitales en lugar de los convencionales como son el papel tendrá una repercusión positiva en el medioambiente.

En el área de la mensajería online, supondrá un impacto muy grande debido a su poco desarrollo para resolver este tipo de problemas como está comentado en el apartado del *estado del arte*. Esta idea está aplicada a los correos electrónicos únicamente, pero se podría de la misma forma aplicar a cualquier otra aplicación de mensajería instantánea o de mensajes en redes sociales como son WhatsApp, Telegram, Instagram, SMS, Twitter… No obstante, esto podría conllevar una pérdida de sincronía en la interacción, ya que un mensaje instantáneo se caracteriza por su rapidez, y esta técnica lo ralentizaría. Como en muchas ocasiones, habría que buscar un balance entre seguridad, usabilidad y rendimiento.

Podría haber consecuencias negativas. Detrás de una buena idea siempre aparece alguien para corromperlas. Los *crackers* o hackers no éticos y delincuentes siempre encuentran la forma de sacar un beneficio sin importarles el daño que causan a las personas. Por ello, no se puede decir que el impacto socioeconómico de este proyecto es 100% positivo debido a los riesgos posibles de fracaso en cuanto encuentren cómo burlar el sistema. Por ejemplo, podrían crear otra entidad de CA y suplantar la de este proyecto. Pero a su vez, hay que tener en cuenta las ventajas considerables, puede que los correos spam dejen de ser un problema, pero surgirán nuevos peligros para la seguridad de los clientes. Podemos decir que el impacto es bastante positivo, pero hay que seguir actualizando y mejorando la seguridad a lo largo del tiempo.

9. CONCLUSIONES Y MEJORAS FUTURAS

El correo electrónico es el canal preferido de comunicación corporativa. Debido a la extensión de la información y datos que viajan a través de internet, la seguridad de internet ha ido decrementando. Los *spammers* son una amenaza bastante común en este campo, enviando de forma masiva correos electrónicos fraudulentos. A pesar de que las empresas también han llevado a cabo medidas preventivas contra las estafas en Internet, no son capaces de controlar completamente todos los mensajes fraudulentos. Sobre todo, la dificultad se halla en la identificación del emisor del mensaje. Por ello, la solución planteada consiste en utilizar una firma electrónica para demostrar su autenticidad.

La solución a este problema se enfoca en la creación de un prototipo de aplicación para identificar los correos de spam y así garantizar que su contenido no se ha modificado ni se ha suplantado la identidad del emisor. Para ello, se aplica un algoritmo de *watermark* o marca de agua como firma digital mejorando la seguridad de los correos electrónicos.

El resultado final cumple con todos los objetivos descritos. El sistema permite detectar los correos ilícitos mediante un algoritmo de *watermarking*. Así, los correos están autenticados y con ello, es posible distinguir entre mensajes legítimos y de spam. Además, se aumenta la confianza de los clientes en la comunicación mediante email con las empresas y viceversa. Por último, para conseguir realizar el sistema propuesto, se ha desarrollado un prototipo compuesto por tres aplicaciones.

Conclusiones personales

Ha sido todo un reto realizar este trabajo de fin de grado debido a no haber visto nada relacionado con la esteganografía durante toda la carrera. Ese hecho se refleja en una investigación previa de varias semanas. En otros proyectos de la carrera apenas tenía que buscar información para entender todos los conceptos, ya que, nos han preparado para ello. En cambio, en este proyecto, había que hacer una investigación exhaustiva en todo momento. A su vez ha sido un aprendizaje muy fructífero.

El resultado es satisfactorio, ya que, considero que refleja el esfuerzo aplicado y demuestra mis conocimientos y competencias. Aunque, se podría mejorar un aspecto de planificación. La redacción de la memoria habría sido menos tediosa si se hubiera redactado a la vez que se implementaba. Si que había notas, pero no estaba bien redactado por lo que podía dar lugar al olvido o confusión de algunos aspectos relevantes.

A nivel profesional, he aprendido que soy capaz de ser todos los roles de un equipo de desarrollo software, por tanto, demostrando el título al que estoy defendiendo de Ingeniera Informática.

A nivel personal, he propuesto una solución que hace un buen uso de la esteganografía, no solo legal sino además solidario con la sociedad. Siempre quise

ayudar a los demás a través de la informática. También, quería demostrarme a mí misma que con actitud y esfuerzo podía aprender de cero conocimientos relacionados con la rama de Ciberseguridad. Es más, este año voy a estudiar los conocimientos necesarios en el Máster de Ciberseguridad en la Carlos III de Madrid que me permitirán extender este proyecto.

Mejoras futuras

Por último, este proyecto podría ampliarse por distintas vías. Por ejemplo, podría hacerse un estudio de la usabilidad, por tanto, mejorando la interfaz de usuario de las aplicaciones. Con esto se podría acercar más a ser un producto final. También, se podrían realizar las contraseñas en hash y así mejorar la seguridad del prototipo.

Por otro lado, se podría mejorar la gestión de las bases de datos, utilizando el lenguaje *Python*. Como se expresó en el estudio tecnológico, *Python* se suele utilizar para aplicaciones de Big Data.

10. PROJECT SUMMARY

This chapter has a summary of this project showing my English skills.

ABSTRACT

Currently, communication among millions of people around the globe is done via email. Both individuals and companies use it. Especially regarding companies, it is important that the exchange of information stays safe, reliable and trustworthy. This fact together with the cyber-attacks evolving and upgrading, for example, starting with spam campaigns, makes it necessary to find a solution to this problem.

This project develops an app to detect spam by verifying the authentication of the sender of the email, using a watermarking algorithm for this. This can solve a major problem, the massive reception of malicious spam emails in companies. For this project, a specific type of watermarking algorithm called Zero-Watermarking is selected. The algorithm used will make sure that every time a user sends an email, there will be a watermark added to the message. The authenticity of the email can be verified thanks to the watermark previously generated. This verification is carried out by an external entity.

Key words: email, spam, authentication, Zero-Watermarking, watermark.

10.1. INTRODUCTION

To define this project, it is important to understand the context, the motivations that have driven this idea and the proposed goals.

10.1.1. Context

The service of electronic mail or email arose even before the appearance of the Internet. In 1965 it began to be used. In 1971 Ray Tomlinson's arroba emerged [1]. This service was essential for communications between users. Despite considerable technological advances, email is still used as the preferred channel of corporate communication [2]. Although social media is used daily by users around the globe, its innovative communication functionality like instant messaging included with many more functionalities than that offered by email, are no rival for conventional online messaging. Users prefer social networks while companies prefer to use email [2].

At first, their security was guaranteed, but as the Internet has expanded and information has become more widespread, spammers have known how to take advantage of the situation. These fraud and deceptive advertising messages are sent daily to customers and employees of companies around the glove. According to a 2016 year-long study by Google [3], it has been discovered that the most common and dangerous cyber-attack on the network is the phishing attack, through deception, they try to extract key user data. The preferred technique is the use of false domains [4] since it is very difficult for the user to detect. There are more and more scams on the Internet and in turn, companies notify their customers of the prevention they should apply and take measures about it, such as updating systems, keeping them virus-free, raising awareness among customers, use strong passwords ... [5]

But just as cyber criminals improve their fraud techniques, companies are not able to control all misleading messages. Just entering your data on a web page is putting your security against spammers at risk. In addition, current instant messaging services are not able to know if the sender of the mail is really who they say they are. A solution proposed by companies is the electronic signature to demonstrate its veracity.

This project focuses on creating an application prototype to detect spam emails. Not only detect them, but also offer a secure channel where companies can guarantee that email has not been tampered with or identity has been impersonated. This prototype uses a watermark algorithm as a signature. It also improves the security of emails in identifying spam messages.

10.1.2. Motivations

This proposal starts with my interest in the cybersecurity branch and the curiosity to explore this field before starting the Cybersecurity Master.

The creation of a watermarking app for spam detection arose from recent complaints of excessive spamming of emails in my social circle. Email managers are not able to detect all spam emails, as these are increasingly unnoticed. In addition, email is widely used throughout the world, especially by companies and their customers. Normally, the exchange of information is confidential. Tampering with or misusing that message could jeopardize the security of both. Therefore, the main motivations of this proposal are to ensure the security and information of emails sent between companies and customers and detect those that are spam, as it is a fairly common problem today where the consequences can be very serious.

10.1.3. Goals

The main goal is to help users identify fraudulent emails and increase their confidence in online messaging systems.

To achieve the previous goal, the search for a cryptographic algorithm is proposed so that the emails are signed, and it is possible to know who has written them. In addition, an implementation of a prototype app to detect functional spam is wanted.

10.2. PREVIOUS KNOWLEDGE

It is necessary to previously explain some relevant concepts.

Steganography, from the Greek **steganos** (hidden) and **graphos** (writing), is the concealment of information in a covert channel, avoiding the discovery of the hidden message [6]. It was not born with computing. It is an art that has been in use since man had the need to hide messages.

Classical steganography is based on ignorance of the specific covert channel being used. Various examples are found throughout history. For example, the animal's or slave's hair was cut, a message was recorded underneath, and the hair was expected to grow to send sensitive information [7]. To decipher the message, the hair was cut, and the hidden message was showed. Also, lemon juice was used on a paper as invisible ink to hide what was written [7]. To decipher it, heat was applied, for example, using a flame generating combustion so that the message appeared. Even during the Second World War, letters were punched in the newspapers which, when exposed to light, letters together formed a message [6].

To better understand how it works, a small example is explained. The key elements are:

- Secret message: information to hide
- Container or concealer object: covered or camouflaged element.

Example: Slave

• Stego-object: container object plus the hidden message.

Example: Slave with hidden message under his hair.

• Guardian: who monitors communication. There are three types: active, passive, or malicious. A passive one tries to discover the algorithm without modifying any object (read only). An active one also modifies the stego-object. A malicious one can do either of the two things described above (not realistic).

• Stego-function: technique for the cover to go unnoticed.

• Stego-key: optional to decipher the information.

In summary, the message travels through an insecure channel and can be intercepted by another person (guardian). The goal is for the guardian to be unaware of the cover, or at least unable to decipher it. To find out, apply the inverse stego-function and the stego-key if it has [7]. All the elements of steganography are related in the scheme of figure 2.1.

The "*Estego-terna*" is made up of three elements: the capacity, quantity of secret information; security, difficulty of detecting the hidden message; robustness, a set of possible modifications before losing all the hidden information [7]. In figure 2.2, there is a graphic representation of the "*Estego-terna*". The more information (quantity), less robustness and less unnoticed (more visible). For a good concealment of the message it is necessary to **balance** the short list.

The current steganography works like the classic one, but it is applied to digital media. There are different types of steganography depending on the type of cover: multimedia content, text, operating systems, and files... In addition, there are different steganography techniques, this document will focus on text *watermarking*.

Watermarking or digital watermark consists of hiding a message in a digital object, in this case in the texts of emails. It is often used to link the author to the object as a technique to seal the authentication.

Authentication is the service that verifies the identity of a user. The digital signature mechanism is used to validate the authenticity and integrity of the message [8]. It is made up of two keys, one public and the other private. With the digital signature, it is shown that the message has not been tampered with and that the sender of the message has been verified [9]. In the digital signature process, a hash function is performed on the document to be signed. A hash function is a mathematical algorithm that transforms one data string into another with a fixed length. Finally, the string returned by the hash function is encrypted using the private key [10]. To decrypt the digital signature, it is necessary to use the public key. By deciphering you can verify the digital signature. The digital signature and verification processes explained are shown in Figure 2.3.

To reinforce the security of this technique, one more element can be used. The measure to improve security chosen, is to include an entity called certificate authority (CA) [13]. A certificate authority (CA) is an external company that validates the authentication of individuals or companies. Two cryptographic keys are generated: one public and the other private. In addition, a *certificate signing request* (CSR) is generated with the public key and relevant company information such as the domain or the email [11]. The CSR is sent to the CA. The CA verifies that the information is correct and digitally signs the certificate with the private key [11]. All these CA functionalities are shown in the diagram in figure 2.4.

10.3. STATE OF THE ART

To explain how the proposed solution has been reached, it is necessary to understand its context, that is, all the previous work that has been done and the similar apps that currently exist. After this study, the proposal can be formalized highlighting its novelty.

10.3.1. Previous works

There are several final degree projects that apply steganography. The most common ones hide a secret message in an image [12]. On the other hand, steganography in texts has been treated more in a theorical than a practical way. Algorithms found in documents have continued to be developed for authentication using steganography [13]. But there is nothing about previous works on spam detection.

On the other hand, there are many scientific articles on spam detection. In one article, spam is detected through the sender-receiver relationships on Twitter [14]. In another, fraudulent messages are detected using correlation, packet symmetry principles, and a statistical method called the "*Sequential Probability Ratio Test*" [15]. Furthermore, spam emails can be identified by *clustering* text based on the vector space model [16].

There are also several scientific articles on watermarking algorithms in texts. In an article, a watermarking algorithm based on the classification of words and statistics of the spaces between words is presented [17]. In another, the text uses a watermarking of the copyright logo image [18]. Finally, a watermarking algorithm that uses the least significant bits (LSB) together with the inverse bits [19].

Finally, to detect spam, a *zero-watermark* algorithm will be used [13]. Despite the existence of scientific articles on spam detection and others of textual watermarking, there are no scientific articles on spam detection using watermarking.

10.3.2. Similar apps

This section shows the apps that are most like the proposed solution for spam detection.

DocuSign

Digital signature app that focuses on authentication and thus avoid fraudulent sources. There is a Chrome extension to be able to sign emails directly from Gmail [20]. In figures 3.1 and 3.2, you can see its logo and its graphic interface.

Mailinblack Protect

Mailinblack Protect is a product aimed at companies and clients that automatically blocks and filters spam emails thanks to its database and servers that work with Artificial Intelligence models. In addition, it has other functionalities such as detecting phishing or spear-phising attempts and unwanted ads. According to Mailinblack, it would take an average of 5 hours for a user to check their spam mail, but with their software tool it would take 40 minutes a day [21]. In figures 3.3 and 3.4, you can see its logo and its graphical interface.

Claranet

Antispam and antivirus services for emails. It offers automatic and guaranteed filtering. It uses real-time spam pattern detection technologies [22]. In figures 3.5 and 3.6, you can see its logo and its services scheme.

10.3.3. Solution novelty

Firstly, as most of the previous works deal with the practical case of hiding data in texts, in this work its app for the detection of spam emails is proposed and it has been decided to use the *zero* algorithm [13], generating a *zero* watermark in the texts of the emails to verify the authentication in a practical way in a software application. The idea itself has not been previously developed, most similarly in theory.

Second, the novelty of the proposal also lies in the set of improved ideas of existing applications. In Table 3.1, they are outlined for better understanding.

By using the apps explained in the previous section, users would save hours identifying fraudulent emails, since it would be an automatic process. Even if only the email inbox were optimized using spam filters and other manual measures available in setup [23].

The suggested solution will save the hours spent detecting spam emails by users. It is not necessary to manually configure the filters as it happens in Gmail. The application most like the proposed solution is *Mailinblack Protect*. Although it has more functionalities and uses more advanced technology, the idea is simpler. It focuses only on detecting spam emails. In addition, using the watermark improves the accuracy of spam email identification. Mailinblack's automatic locks and filters may fail [24] but

signing with the watermark is less error-prone because each email is being checked individually. As the watermark is a technique of steganography, it is less detectable than the DocuSign application digital signature.

In summary, the saving of hours for the user due to its automatic process, its simplicity, and its approach to security, make this idea a proposal with significant potential. Furthermore, no application has ever been made using steganography to identify spam emails.

10.4. ANALYSIS

10.4.1. Proposed solution

After concluding that the novel idea of the project does not appear in any previous works or projects, the solution to the problem is proposed.

To send an email two elements are needed: sender and receiver. The sender will be the company and the receiver will be the customer. In addition, an intermediary entity is required to verify that the email sent belongs to the original sender and is not spam.

More specifically, it is necessary to create three software apps. The application for companies will be called "AquaSpam". AquaSpam's main service will be to send an email to customers with the watermarking signature. The customer app will be called “Gmail Add-on AquaSpam” and will be available in some way within Gmail to be able to extract the information and send a verification request to the intermediary app. The intermediary app called "CA", will have two databases. A database will store the watermarks created every time a company sends an email to its customer. The other database will store the verification demand requested by the customer. In addition, in the CA app, each request can be verified by recalculating the watermark and comparing it with those stored in the watermark database. Finally, the results will be sent by mail and will be removed from the corresponding databases to save memory space. All this is outlined in detail in Figure 4.1.

10.4.2. Software requirements

To develop any software application, it is essential to specify the functionalities of the system together with its restrictions. All this is included in the Software Requirements Specification (SRS). There are two types of fundamental requirements: functional and non-functional.

Functional requirements describe how the system works and services of the system. They must be complete and consistent, so they meet all customer expectations.

Non-functional requirements define the restrictions of the system, functions, or its properties such as performance, security, and availability.

All the software requirements have been taken for each app: AquaSpam (Company App), Gmail Add-on AquaSpam (Customer App) y CA (Intermediary App).

The two most important requirements for AquaSpam app are to let the user to log in (RFA-01) and being able to send a message to its costumer via email with a watermark signature (RFA-03). In Gmail Add-on AquaSpam app, the possibility to send a verification request is the main requirement (RFB-04). The two most significant requirements for CA app are to let the user to log in (RFC-01) and being able two verify the request asked for (RFC-09).

10.5. DESIGN

The system is made up of three apps, as indicated in figure 5.1. The company's email is used as a database to store both the generated watermarks and verification requests. Companies use AquaSpam app to send emails with a message and its watermark to their customers. In turn, this watermark is stored in the company's email together with the id, that is, the date and time of email creation. It is stored under the subject "Watermak Created & Stored." Clients use Gmail Add-on AquaSpam to request verification. The request is stored in the same company email under the subject "Verification request for CA". AquaSpam technicians or workers query the databases using the CA app. Also, they use the app to run the verification. Automatically, the results of the algorithm are sent and finally, they are deleted from the databases to save space.

10.6. IMPLEMENTATION

First, the algorithm used is exposed and then, the keys of the three apps are briefly explained. To see it in practice, a small guide through the three apps is attached in the *Anexo: Manual de Usuario* section.

10.6.1. Algorithm

The algorithm used is called "Zero-Watermarking" [13]. It consists of given a plain text and the selection of a keyword (kw), a watermark is generated to identify the ownership of the file to which the algorithm is applied, in this case who sends the message.

For keyword selection, the algorithm chooses the most frequent word from the text. To create the watermark, all the words in the text that match the chosen key are searched in order. The letters of the words immediately before and after the key found are counted. Therefore, the watermark has the keyword and a sequence of numbers. This first process is called "Embedding". In figure 6.1, it is shown visually how a watermark is generated.

Then an external entity called the Certifying Authority or Certificate Authority (CA) stores the watermark along with an id. The id used is the date and time in “DD / MM / YY HH: MM: SS” format.

The last process is the extraction process. It is identical to embedding, but it is done on the received message, not on the sent one. It is performed to verify that the message has not been tampered with and that the author of the message has been verified correctly. Finally, the CA storage is accessed to verify that the watermarks match. In figure 6.2, you can see the general scheme of the algorithm.

10.6.2. Apps

After exposing the algorithm, the implementation of each of the proposed apps is exposed.

First, the company decides to send an email through the AquaSpam app. Then, the “Embedding” algorithm is executed and the message is subsequently sent with the corresponding watermark. Figure 6.3 shows the AquaSpam app screens.

Second, the customer makes a request for verification of the mail sent by the company. It is made through Gmail using the Gmail add-on from AquaSpam. Figure 6.4 shows the screens of the AquaSpam Gmail add-on application.

Finally, the CA application accesses the company email created by and for AquaSpam (technologysa.aquaspam@gmail.com) and thus manage all companies individually (simplification of the prototype). In that email, there are two databases, one stores the watermarks (WatermarkDatabase) and the other stores the requests (RequestsDatabase). AquaSpam employees would be responsible for verifying those requests. The verification algorithm would be the one previously called "extraction". This process can be automated, but the proposed development is manual to see all the processes in detail. Figure 6.5 shows the CA app screens.

For logins in the AquaSpam and CA apps, it is necessary to use a password. These passwords would have to be hashed in the databases to avoid password theft if the databases were compromised. This functionality has not been implemented because it is a prototype and there were elements of greater relevance. But this does not mean that it was not considered.

10.7. EVALUATION

The tests designed are defined in table 7.1. They verify that the apps do what they are supposed to do. As can be seen in Table 7.2, a good analysis and design have been done, since the results show that all acceptance tests have been **passed**.

10.8. CONCLUSIONS AND FUTURE IMPROVEMENTS

Email is the preferred channel for corporate communication. Due to the widespread of information and data transmission through the internet, internet security has been decreasing. *Spammers* are a common threat in this field, sending fraudulent emails in bulk. Even though companies have also taken preventive measures against internet scams, they are not able to fully control all fraudulent messages. Above all, the difficulty lies in identifying the sender of the message. Therefore, the proposed solution is to use an electronic signature to demonstrate its authenticity.

The solution to this problem focuses on creating an application prototype to identify the spam emails and thus guarantee that their content has not been modified or the identity of the sender has been supplanted. To do this, a *watermark* algorithm is applied as a digital signature, improving the security of emails.

The result meets all the objectives described. The system allows detecting illegal emails using a *watermarking* algorithm. So, the emails are authenticated and with it, it is possible to distinguish between legitimate and spam messages. In addition, customer confidence in email communication with companies and vice versa is increased. To do the proposed system, a prototype composed by three apps has been developed.

Personal conclusions

It has been a challenge to do this final degree project as I did not see anything related to steganography during the entire degree. That fact is reflected in a previous investigation of several weeks. In other projects of the degree, I hardly had to look for information to understand all the concepts, since they have prepared us for it. Instead, in this project, exhaustive research always had to be done. It has been a very useful learning.

The result has been satisfying, since I consider that it reflects the applied effort and demonstrates my knowledge and skills. Although, one aspect of planning could be improved. The writing of the report would have been less tedious if it had been written at the same time as it was implemented. The notes written could lead to the forgetfulness or confusion of some relevant aspects.

At a professional level, I have learned that I am capable of being all the roles of a software development team, therefore, proving the title that I am defending as a Computer Engineer.

On a personal level, I have proposed a solution that makes good use of steganography, not only legal but also giving aid to society. I always wanted to help others through computing. Also, I wanted to show to myself that with attitude and effort I could learn about Cybersecurity without any previous knowledge. This year I am going to study the necessary knowledge in the Cybersecurity Master at Carlos III in Madrid that will allow me to extend this project.

Future improvements

Finally, this project could be expanded in different ways. For example, a usability study could be done, therefore, by improving the user interface of apps. It could be closer to being a final product. Also, the passwords could be made in hash and thus improve the security of the prototype.

On the other hand, the management of the databases could be improved, using the *Python* language. As expressed in the technology study, *Python* is often used for Big Data applications.

11.BIBLIOGRAFÍA

[1] R. Martín, "El Correo Electrónico: Origen y Funcionamiento", TecnoCosas. Recuperado de: https://www.tecnocosas.es/el-correo-electronico-origen-y-funcionamiento/ (18 de abril de 2009)

[2] K. Naragon, "Email – Still the Alpha Channel", Adobe Blog. Recuperado de: https://theblog.adobe.com/email-still-the-alpha-channel/ (10 de marzo de 2016)

[3] M. Moon, "Google study shows how your account is most likely to be hijacked", Engadget. Recuperado de: https://www.engadget.com/2017-11-11-google-study-hijack.html (11 de noviembre de 2017)

[4] A. Núñez-Torrón Stock, "9 de cada 10 dominios falsos hallados en 2018 sigue en activo", Ticbeat. Recuperado de: https://www.ticbeat.com/seguridad/9-de-cada-10-dominios-falsos-hallados-en-2018-sigue-en-activo/ (4 de julio de 2019)

[5] INCIBE, Instituto Nacional de Ciberseguridad. "Protege a tus clientes". Recuperado de: https://www.incibe.es/protege-tu-empresa/que-te-interesa/protege-tus-clientes

[6] INTECO, Instituto Nacional de Tecnologías de la Comunicación. "Cuaderno de notas del observatorio: Esteganografía, el arte de ocultar información". Recuperado de: https://pgarciab.firstcloudit.com/documentos/esteganografia.pdf

[7] P. F. Iglesias, "#Mundohacker: Esteganografía, el arte de ocultar información sensible", PabloYglesias. Recuperado de: https://www.pabloyglesias.com/mundohacker-esteganografia/ (28 de abril de 2015)

[8] Sede Electrónica Real Casa de la Moneda, Fábrica Nacional de La Moneda y Timbre, "1032-¿Qué significa Autenticación?". Recuperado de: https://www.sede.fnmt.gob.es/preguntas-frecuentes/otras-preguntas/-/asset_publisher/1RphW9IeUoAH/content/1032-que-significa-autenticacion-?inheritRedirect=false#:~:text=La%20autenticaci%C3%B3n%20es%20un%20servicio,y%20por%20tanto%20su%20autenticidad.

[9] L. Soto, "¿Qué es una firma digital?", Signaturit. Recuperado de: https://blog.signaturit.com/es/que-es-una-firma-digital (19 de junio de 2018)

[10] V. Iglesias, "Cómo funciona la firma digital", VictorIglesias. Recuperado de: https://www.victoriglesias.net/como-funciona-la-firma-digital/ (4 de agosto de 2017)

[11] A. Russell, “What is a Certificate Authority (CA)?”, SSL. Recuperado de: https://www.ssl.com/faqs/what-is-a-certificate-authority/ (23 de julio de 2019)

[12] S. Pérez Olivares, “Desarrollo de una aplicación esteganográfica para Android”, Trabajo Fin de Grado, Dpto. de Informática, Universidad Carlos III de Madrid, Colmenarejo, España, 2013. [En línea]. Disponible en: http://hdl.handle.net/10016/23674

[13] Z. Jalil, A. M. Mirza y M. Sabir, “Content based Zero-Watermarking Algorithm for Authentication of Text Documents”, *(IJCSIS) International Journal of Computer Science and Information Security*, Vol. 7, No.2, pp.212-217, February 2010.

[14] J. Song, S. Lee y J. Kim, “Spam Filtering in Twitter Using Sender-Receiver Relationship”, Lecture Notes in Computer Science, vol 6961. Springer, Berlin, Heidelberg, 2011.

[15] M. Xie, H. Yin y H. Wang, “An effective defense against email spam laundering”, Association for Computing Machinery, New York, NY, USA, 179–190, 2006.

[16] M. Sasaki y H. Shinnou, "Spam detection using text clustering", International Conference on Cyberworlds (CW'05), Singapore, pp. 4 pp.-319, 2005.

[17] Y. W. Kim, K. A. Moon y I. S. Oh, “A Text Watermarking Algorithm based on Word Classification and Inter-word Space Statistics”. Seventh International Conference on Document Analysis and Recognition. Proceedings., Edinburgh, UK, pp. 775-779, 2003

[18] Z. Jalil y A.M. Mirza, “An Invisible Text Watermarking Algorithm using Image Watermark”. Innovations in Computing Sciences and Software Engineering. Springer, Dordrecht, 2010.

[19] Bamatraf, A., Ibrahim, R., Salleh, M., & Mohd, N., “A new digital watermarking algorithm using combination of least significant bit (LSB) and inverse bit”, 2011

[20] DocuSign, “Portada” . Recuperado de: https://www.docusign.com/

[21] Mailinblack, “Productos: Mailinblack Protect”. Recuperado de: https://www.mailinblack.com/es/producto/mailinblack-proteccion-anti-spam/

[22] Claranet, “Servicios: Seguridad Antispam y Antivirus”. Recuperado de: https://www.claranet.es/servicio-de-antispam-y-antivirus-profesional-para-empresas

[23] M. Cid, “Inundado de emails: cómo gestionar tu Gmail para evitar perder tu día con spam”, El Confidencial: Tecnología. Recuperado de: https://www.elconfidencial.com/tecnologia/2019-12-24/gmail-bandeja-entrada-google-organizar-perder_2388572/ (26 de diciembre de 2019)

[24] BBC News Mundo, "Cuáles son los principales archivos con los que puedes recibir un virus en tu correo electrónico", BBC. Recuperado de : https://www.bbc.com/mundo/noticias-48800245 (28 de junio de 2019)

[25] B. Santana, "Lenguajes más usados en 2019", 2coders. Recuperado de: https://www.2coders.com/lenguajes-mas-usados-en-2019/ (17 de enero de 2020)

[26] D. Zomaya, "Python vs Java: Which Programming Language is Right for You?", Udemy. Recuperado de: https://blog.udemy.com/python-vs-java/ (marzo 2020)

[27] S. J. Vaughan-Nichols, "The battle over the universal Java IDE," in *Computer*, vol. 36, no. 4, pp. 21-23, April 2003.

[28] Google Inc., "Sitios web y aplicaciones de terceros con acceso a tu cuenta", Google Support. Recuperado de: https://support.google.com/accounts/answer/3466521?hl=es

[29] Grupo Vadillo, "[Guía] ¿Qué aspectos legales debe cumplir una página web? ", issuu. Recuperado de: https://issuu.com/grupovadillo/docs/dsi-requisitos_legales_web-vadillo (19 de mayo de 2019)

[30] Ayuda Ley Protección Datos, "¿Qué es la Agencia Española de Protección de Datos (AEPD)?". Recuperado de: https://ayudaleyprotecciondatos.es/2010/05/03/que-es-la-agencia-espanola-de-proteccion-de-datos-aepd/

[31] J. Miquel Serrano. *¿Qué es la LOPD? Nueva normativa protección de datos 2018*. (20 julio de 2017). Acceso: 17 de junio de 2020. [Video en línea]. Disponible en: https://youtu.be/BRdrgBYjxm4

[32] Política de Cookies, "Concepto de Cookie". Recuperado de: http://politicadecookies.com/index.php

[33] European Data Protection Supervisor, "ePrivacy Directive". Recuperado de: https://edps.europa.eu/data-protection/our-work/subjects/eprivacy-directive_en

[34] Privazy Plan, "Artículo 6 UE RGDP: Licitud del tratamiento". Recuperado de: https://gdpr.algolia.com/es/gdpr-article-6

[35] Reglamento General de Protección de Datos, "Nuevo Reglamento - Introducción". Recuperado de: https://rgpd.es/

[36] LinkedIn, "Descubre tu remuneración potencial". Recuperado de: https://www.linkedin.com/salary/ (23 de junio de 2020)

12.ANEXO

12.1. Manual de Uso

En esta sección se pretende realizar una pequeña guía de uso para facilitar la navegación al usuario por las tres aplicaciones implementadas. Se necesitarán dos correos, uno que representa a la empresa (*technologysa.aquaspam@gmail.com*) y otra al cliente (*sofianajalm@gmail.com*).

12.1.1. AquaSpam

Aplicación para empresas. Las empresas pueden enviar mensajes con *watermarking* o marcas de agua a sus clientes.

Instalación y requisitos previos

Únicamente es necesario instalar la máquina virtual de Java (si no estaba instalada) para poder abrir cualquier archivo con extensión *jar*. A continuación, abrir el archivo "*AquaSpam.jar*". Este archivo ejecuta todo el código implementado.

Ejecución

En esta sección se muestran los pasos a seguir para navegar por AquaSpam:

- Paso 1 -La primera pantalla es de bienvenida. Para continuar, se presiona con el ratón en la pantalla como se indica por pantalla (fig. 33).

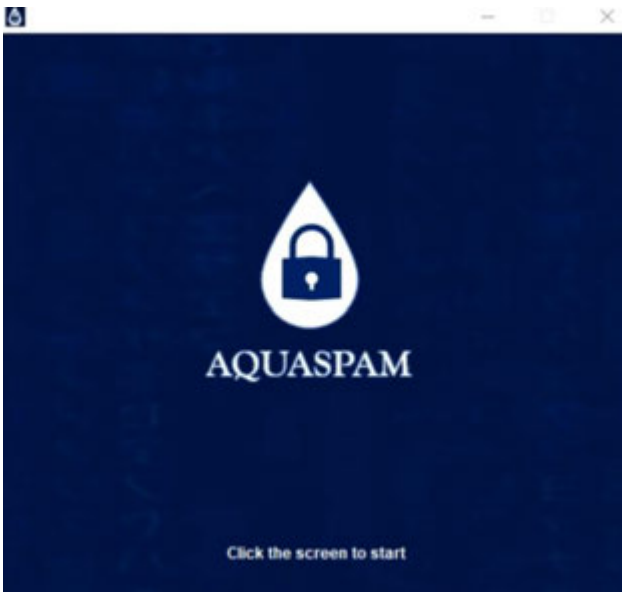


Fig. A.1. Paso 1- AquaSpam

- Paso 2- Se muestra una pantalla de *login* o inicio de sesión. Al ser un prototipo no hay ventana de ingreso o registro, es decir, se da por hecho que

este proceso ya se ha realizado previamente. El usuario tiene que introducir el correo electrónico específico creado por AquaSpam para la empresa y una contraseña (fig. 34).

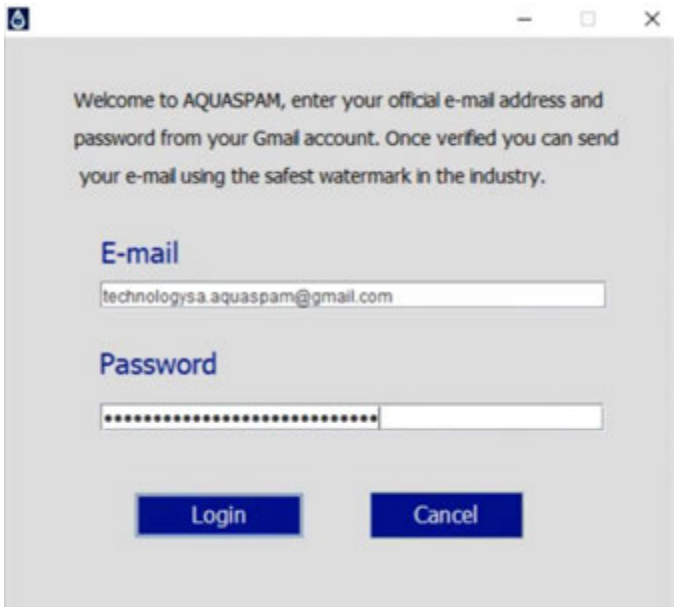


Fig. A.2. Paso 2- AquaSpam

- Paso 3- Una vez iniciada la sesión, aparece una ventana para crear y enviar un nuevo correo a un cliente. Primero, se introduce el correo del cliente al que se le quiere enviar el correo (*To*). Después, se añade un asunto (*Subject*). Finalmente, se redacta el mensaje que se quiere enviar (fig. 35).

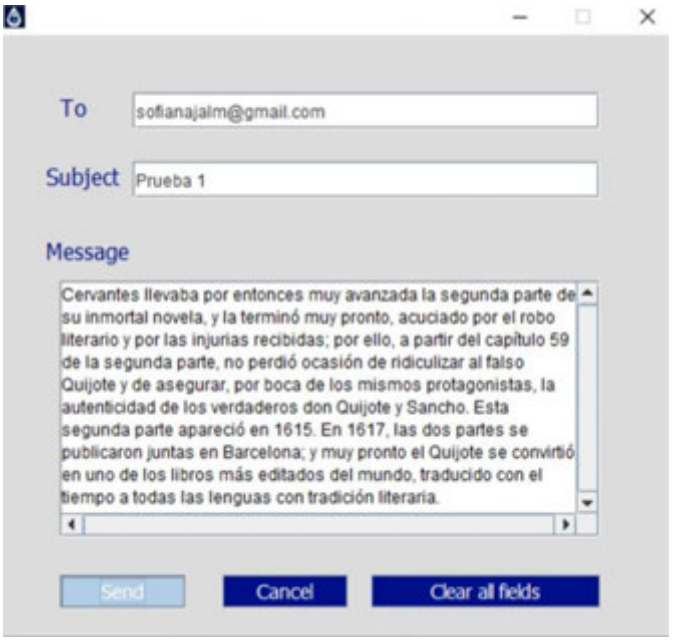


Fig. A.3. Paso 3- AquaSpam

- Paso 4- Si no se quisiera enviar ese mensaje se puede pulsar la casilla "*Cancel*". Si se quisieran borrar todos los campos, se pulsa "*Clear all fields*". Al darle al botón de enviar (*Send*) la aplicación internamente añade la marca de agua al texto enviado para el cliente (fig. 36).

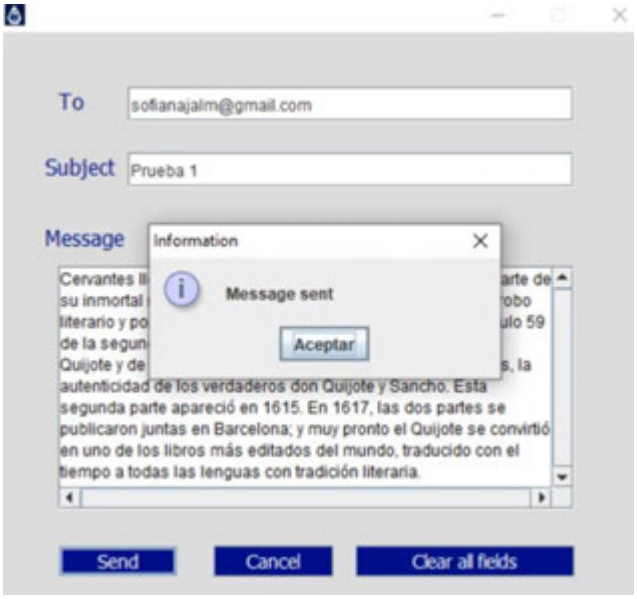


Fig. A.4. Paso 4- AquaSpam

- Paso 5- Se comprueba que se ha enviado y se ha creado la marca de agua correctamente. Para ello, se entra en el correo electrónico de la empresa donde se almacena el *watermark*. Mirar correos con asunto "*Watermark Created & Stored*" (fig. 37).

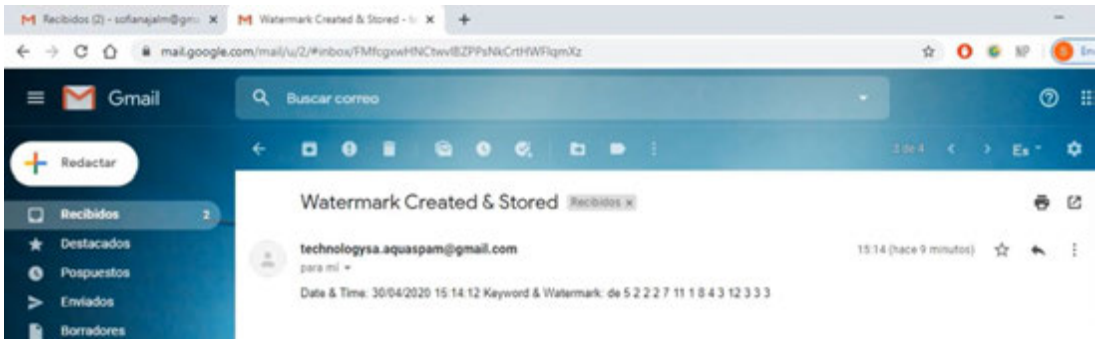


Fig. A.5. Paso 5- AquaSpam

12.1.2. Gmail Add-on AquaSpam

Aplicación para clientes. Los clientes pueden pedir la verificación de un correo específico enviado por una empresa registrada en AquaSpam. La verificación se envía a la siguiente aplicación de la guía: CA.

Instalación y requisitos previos

El correo del cliente tiene que ser de *Gmail*. Si este plugin estuviera disponible en *G Suite Marketplace*, simplemente hay que seleccionar el signo de más "+" en la parte derecha de la ventana "*Descargar complementos*". Si no fuera así, se trata de una aplicación no publicada (unpublished add-on) o de desarrolladores (developer add-on) y habría que seguir los siguientes pasos:

A- Ir a la cuenta Google Drive donde está el código. Se trata de un archivo *gs* y de un manifiesto llamado "*appsscript.json*". Para que aparezca el manifiesto, hay que ir a "*Ver*" y luego a "*Mostrar archivo de manifiesto*".

B- Para publicar la aplicación y poder instalarla, hay que ir a "*Publicar*" y a "*Desplegar del manifiesto*". Se abre una ventana llamada "*Deployments*" y se instala el complemento. Si se realizara cualquier cambio al código se actualiza automáticamente sin necesidad de reinstalarlo. Este paso se muestra en la figura 38.

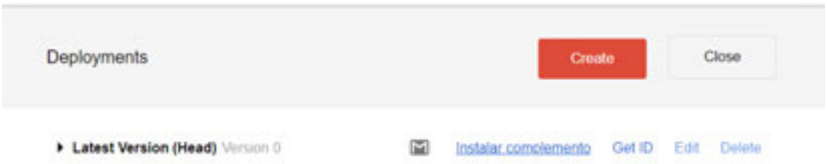


Fig. A.6. Paso B- Gmail Add-on

C- Finalmente, se comprueba que aparece el complemento descargado. Tiene que verse en cualquier ventana del correo en la parte derecha y al ir a "*Configuración*" en "*Complementos*". Este paso se muestra en la figura 39.

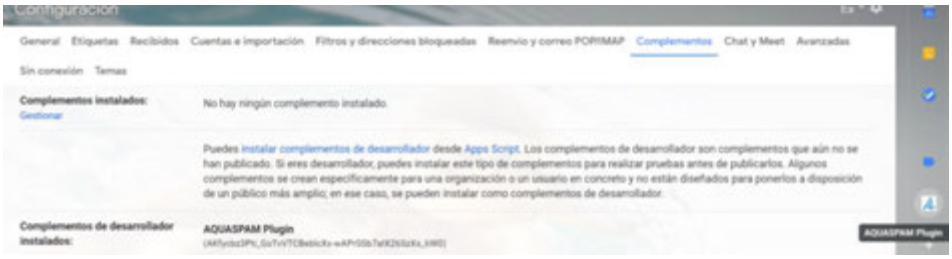


Fig. A.7. Paso C- Gmail Add-on

Ejecución

En esta sección se muestran los pasos a seguir para navegar por el Gmail Add-on AquaSpam:

- Paso 1- Se entra en el correo electrónico. Una vez se abre, pulsar el icono del plugin (fig. 40).

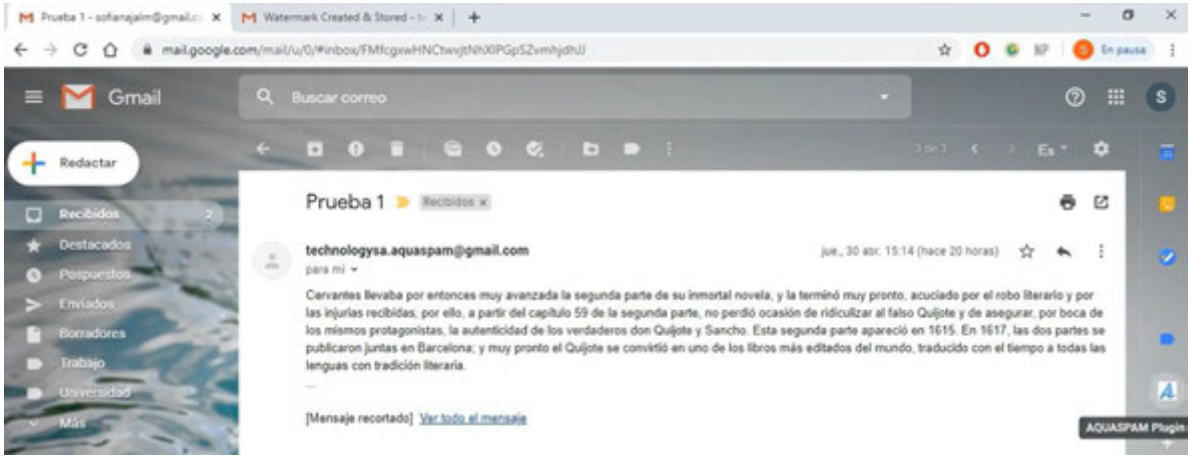


Fig. A.8. Paso 1- Gmail Add-on

- Paso 2- Aparece en la parte derecha una pequeña ventana del plugin. Se muestra la extracción del texto del correo, es decir, la fecha, la hora y el mensaje. En la parte inferior, hay un botón "*Verify*" para enviar una solicitud de verificación a la siguiente aplicación *CA* con los datos extraídos (fig. 41).

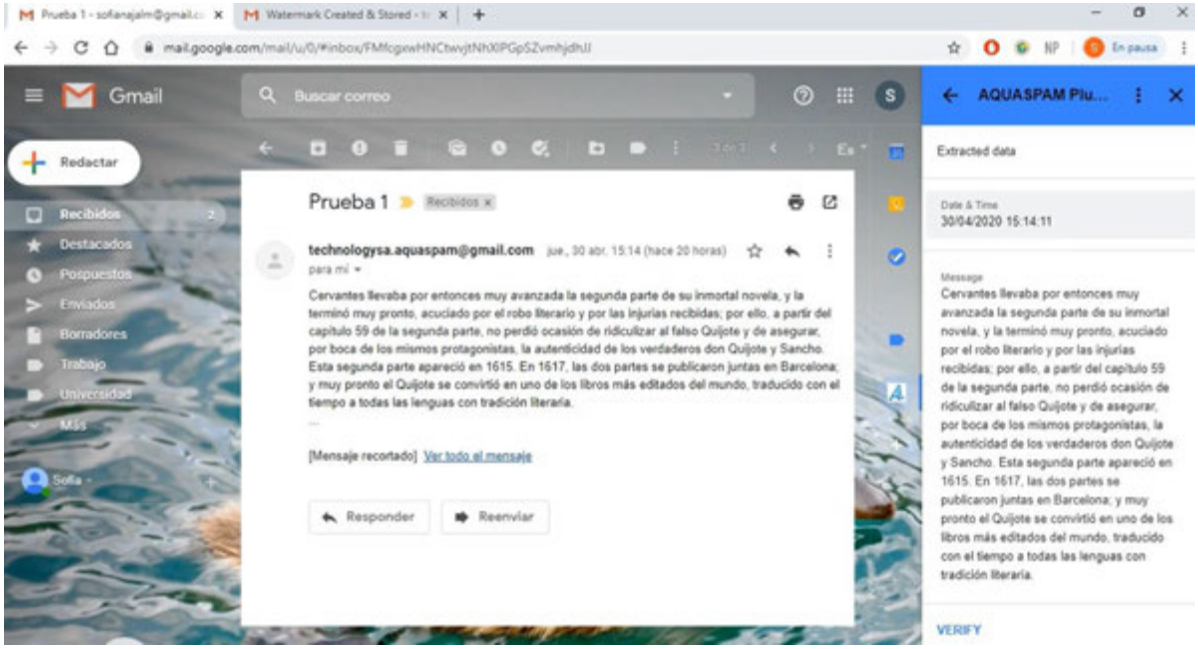


Fig. A.9. Paso 2- Gmail Add-on

- Paso 3- Se muestra una caja de información de que el mensaje se ha enviado correctamente (fig. 42).

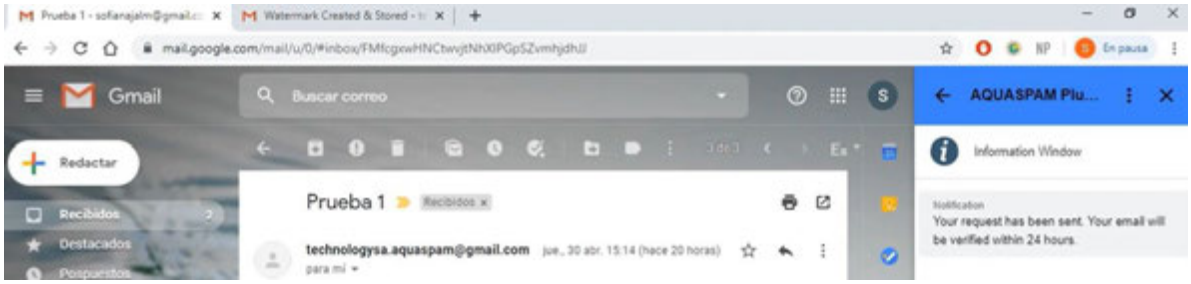


Fig. A.10. Paso 3- Gmail Add-on

- Paso 4- Se comprueba que se ha realizado la petición correctamente. Para ello, se entra en el correo electrónico de la empresa donde se almacena la petición. Mirar correos con asunto "*Verification request for CA*" (fig. 43).

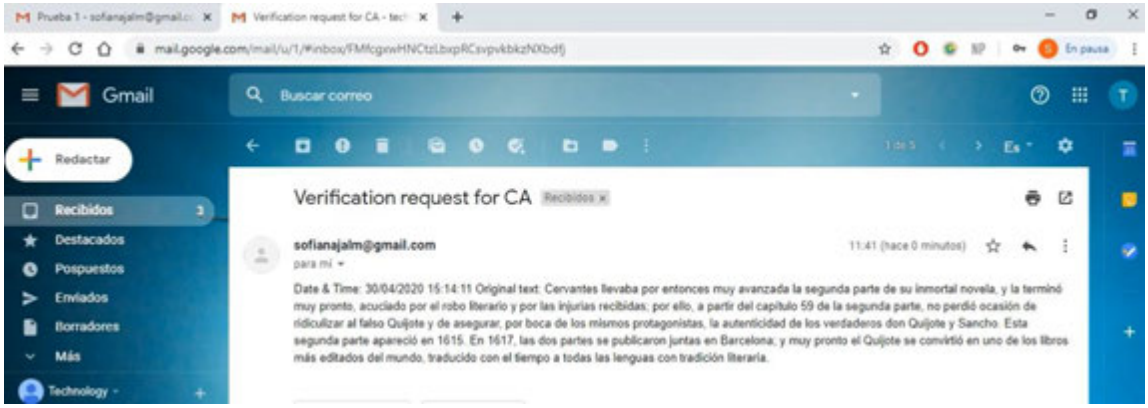


Fig. A.11. Paso 4- Gmail Add-on

12.1.3. CA

Aplicación para los técnicos IT de AquaSpam, sería la aplicación intermediaria entre empresas y clientes. Llamada *CA* por su nombre "*Certificate Authority*". Se puede acceder a la lista de empresas. Cada empresa tiene dos bases de datos, una donde se almacenan las marcas de aguas y otra donde se almacenan las peticiones de los clientes para verificarlas. Se pueden observar las bases de datos y aceptar las solicitudes de verificación. Tras la verificación, se enviarían los correos automáticamente con el resultado.

Instalación y requisitos previos

Únicamente es necesario instalar la máquina virtual de Java (si no estaba instalada) para poder abrir cualquier archivo con extensión *jar*. A continuación, abrir el archivo "*CA.jar*". Este archivo ejecuta todo el código implementado.

Ejecución

En esta sección se muestran los pasos a seguir para navegar por CA:

- Paso 1- La primera pantalla es de bienvenida. Para continuar, se presiona con el ratón en la pantalla como se indica por pantalla (fig. 44).

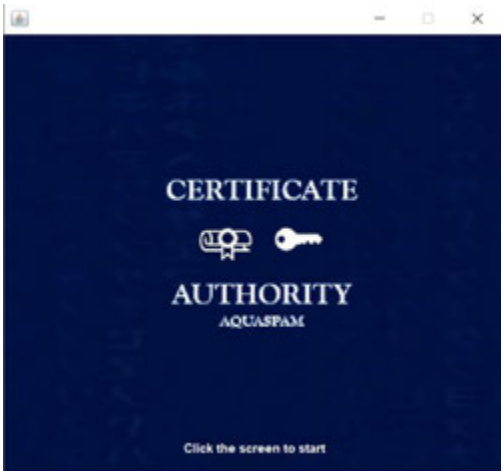


Fig. A.12. Paso 1- CA

- Paso 2- Se muestra una pantalla de *login* o inicio de sesión para trabajadores de AquaSpam. El usuario tiene que introducir el id de trabajador y una contraseña (fig. 45).

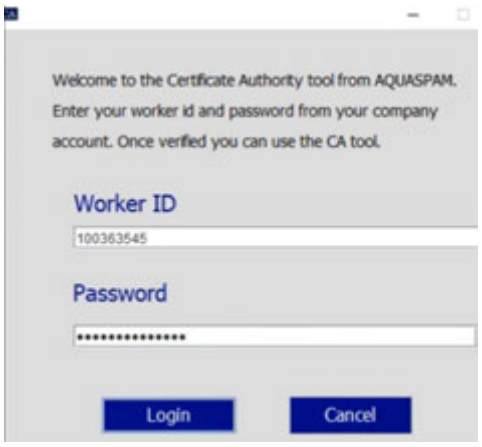


Fig. A.13. Paso 2- CA

- Paso 3- Se muestra una pantalla con la lista de empresas. En la parte superior, se indica el perfil del trabajador. En la esquina superior derecha hay un botón para cerrar sesión. Se selecciona la empresa "*Technology S.A.*" (fig. 46).

Fig. A.14. Paso 3- CA

- Paso 4- Se muestra una pantalla con las dos bases de datos. Primero, se selecciona la base de datos de las marcas de agua: "*Watermarks Database*" (fig. 47).

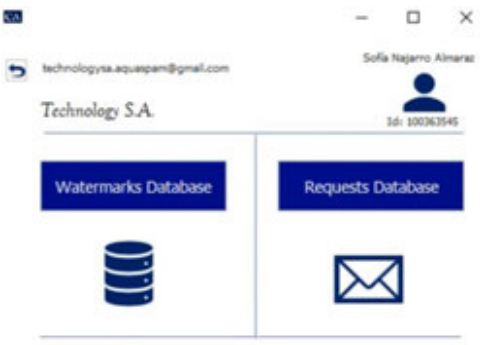


Fig. A.15. Paso 4- CA

- Paso 5- Las marcas de agua se componen de tres elementos: id (fecha y hora), kw (*keyword* o clave que es la palabra más repetida) y watermark (secuencia de números). Para ver cada elemento hay que pulsar las pestañas superiores. Se lee en horizontal. Si se sigue la Figura 48 y las líneas sombreadas, la primera marca de agua sería: "*02/05/2020 10:47:11 de 5 2 2 2 7 11 1 8 4 3 12 3 3 3*". Una vez visualizada, se vuelve a la pantalla anterior pulsando el botón de la esquina superior izquierda con una flecha como símbolo.

Fig. A.16. Paso 5- CA

- Paso 6- Se vuelve a mostrar la pantalla con las dos bases de datos. Ahora, se selecciona la base de datos de las peticiones de verificación: "*Requests Database*".

Fig. A.17. Paso 6- CA

- Paso 7- Las peticiones se componen de dos elementos: id (fecha y hora) y el texto original. Para ver cada elemento hay que pulsar las pestañas superiores. Se lee en horizontal. Para desplazarse por cada petición hay unos botones de navegación de flechas en la parte inferior derecha e izquierda. Una vez seleccionada una petición, se pulsa el botón "*Verify*".

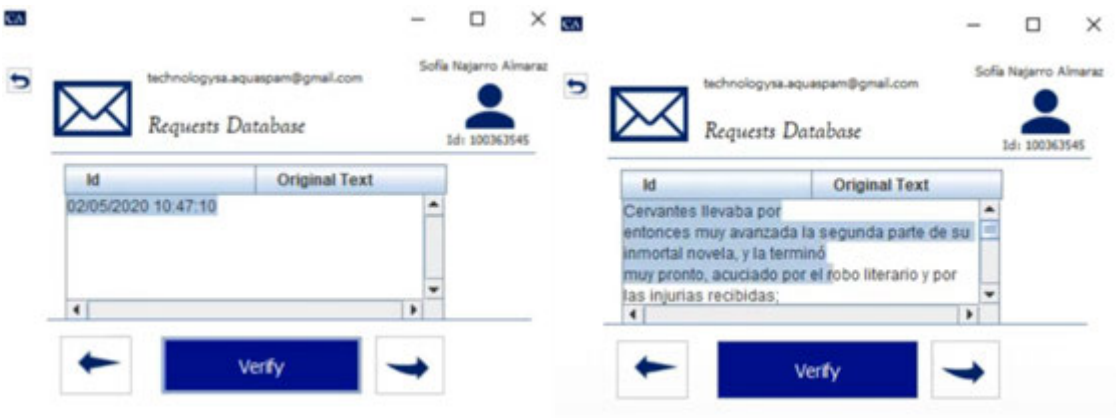


Fig. A.18. Paso 7- CA

- Paso 8- El proceso de verificación se divide en 3 fases o *steps*. El algoritmo se realiza internamente, la aplicación únicamente muestra las acciones realizadas, si ha habido error o se han realizado correctamente. En la primera fase, se comprueba que el id de la petición coincide con algún id de las bases de datos de marcas de agua. Si se procede con éxito, aparecen ventanas informativas. Para continuar, pulsar el botón "*Aceptar*".

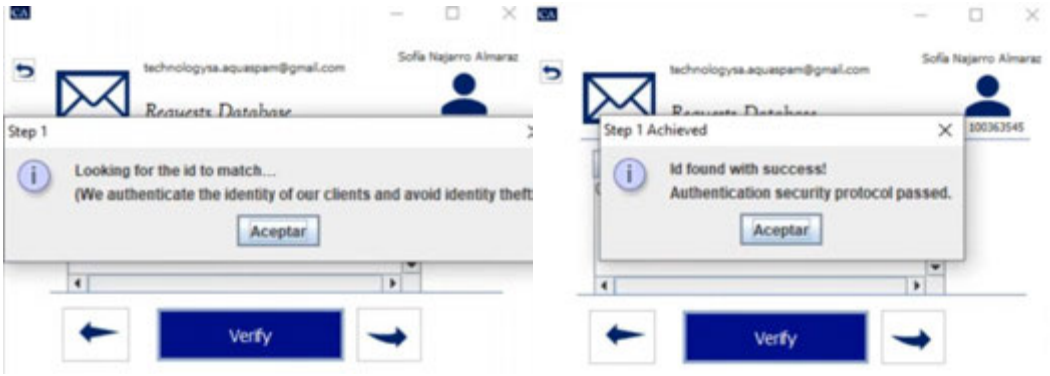


Fig. A.19. Paso 8- CA

- Paso 9- En la segunda fase, se calcula la clave o *kw* y la marca de agua con el mismo algoritmo que el usado en la aplicación AquaSpam con el texto original de la petición. Si se procede con éxito, aparecen ventanas informativas. Para continuar, pulsar el botón "*Aceptar*".

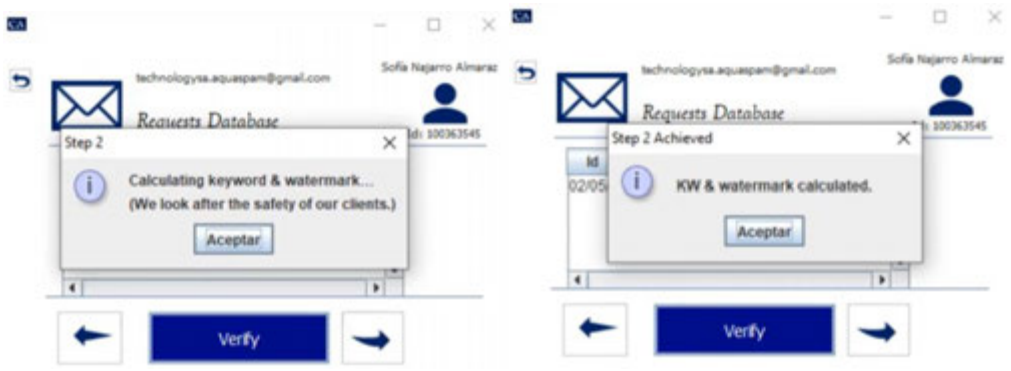


Fig. A.20. Paso 9- CA

- Paso 10- La tercera fase se divide en dos subfases. Primero, se comprueba que coindicen los kw. Si se procede con éxito, aparecen ventanas informativas. Para continuar, pulsar el botón "*Aceptar*".

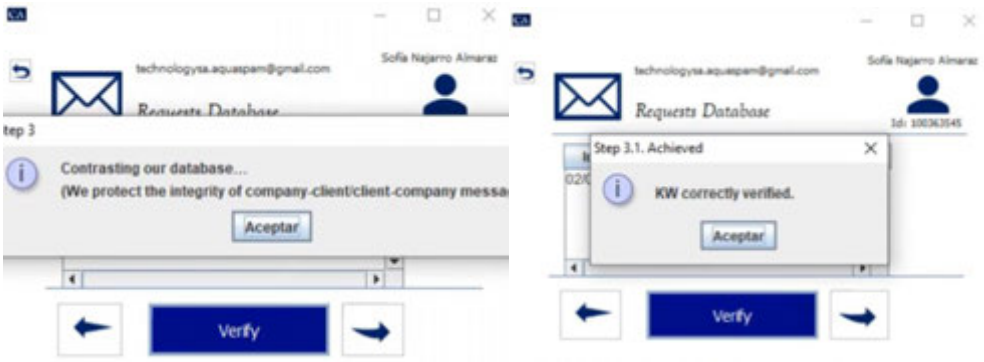


Fig. A.21. Paso 10- CA

- Paso 11- Segundo, se comprueba que coindicen las marcas de agua. Si se procede con éxito, aparecen ventanas informativas y se envía un mensaje al cliente. Para continuar, pulsar el botón "*Aceptar*".

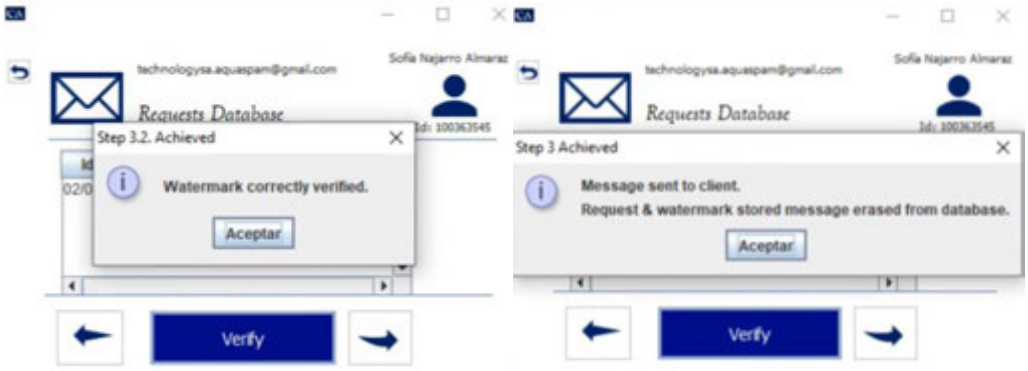


Fig. A.22. Paso 11- CA

- Paso 12- Se comprueba que se ha realizado la verificación correctamente. Para ello, se entra en el correo electrónico del cliente. Mirar correos con asunto: "*SUCCESS: Verification request for CA*". Si ha habido algún error en el proceso se buscan correos con asuntos: "*FAILED: Verification request for CA*" y a su vez, se envían correos a la empresa con el asunto: "*SECURITY PROBLEM: Verification request for CA*".

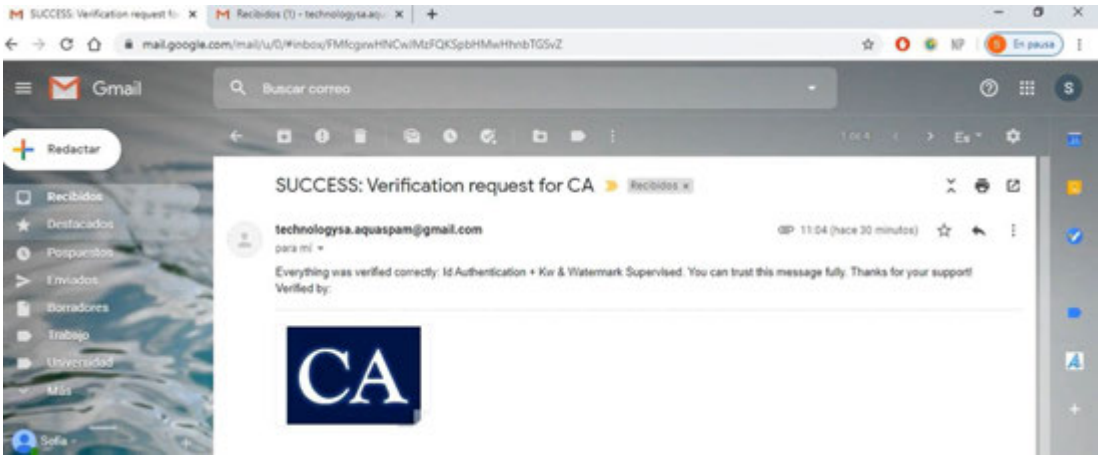


Fig. A.23. Paso 12- CA

12.2. Plantillas

12.2.1. Plantilla para la especificación de requisitos de software

TABLA A.1. PLANTILLA PARA LA ESPECIFICACIÓN DE REQUISITOS DE SOFTWARE

Requisitos de Software de la app Z				
Tipo: Funcional o no funcional				
Categoría: Grupo representativo para encontrarlos con mayor facilidad.				
Id	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RFX-Y o *RNFX-Y,* donde RF (requisito funcional) y RNF (requisito no funcional). Además, Y indica el número del requisito y X indica la aplicación a la que se refiere, pueden ser A (AquaSpam), B(Gmail Add-on) o C (CA).	El nombre del requisito.	Explicación detallada del requisito.	Indica como de estable es, es decir, que no haya sido modificado durante el resto del proyecto. Puede ser alta, media o baja.	Indica la importancia que tiene el requisito para los clientes y usuarios. Puede ser alta, media o baja.

12.2.2. Plantilla para los casos de uso

TABLA A.2. PLANTILLA PARA LOS CASOS DE USO

Id: *CU-Y,* donde CU (caso de uso) e Y indica el número de caso de uso.
Nombre: Palabra o frase representativa que lo identifica.
Descripción: Explicación detallada del caso de uso.
Actores: Alguien o algo externo que interactúa con el sistema**.**
Precondiciones: Condiciones previas para poder ejecutar el caso de uso.
Postcondiciones: Condiciones posteriores al ejecutar el caso de uso**.**
Flujo Normal: Conjunto de acciones necesarias para llegar a un resultado**.**

12.2.3. Plantilla para las pruebas de aceptación

TABLA A.3. PLANTILLA PARA LAS PRUEBAS DE ACEPTACIÓN

Pruebas de aceptación			
Aplicación: App Z			
Id	**Requisitos**	**Entrada**	**Salida**
PAX-Y, donde PA (prueba de aceptación). Además, Y indica el número de la prueba y X indica la aplicación a la que se refiere, pueden ser A (AquaSpam), B(Gmail Add-on) o C (CA).	Los requisitos que aparecen en esta prueba.	El escenario de la prueba a realizar.	El resultado tras realizar la prueba.